HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

Analyse, Konzeption und prototypische Umsetzung einer prozessorunabhängigen XSL-FO Entwicklungsumgebung

Daniel Merkle

Konstanz, 16.01.2008

DIPLOMARBEIT

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Informatiker (FH)

an der

Hochschule Konstanz

Technik, Wirtschaft und Gestaltung

Fakultät Informatik
Studiengang WI

Thema: **Analyse, Konzeption und prototypische Umsetzung einer prozessorunabhängigen XSL-FO Entwicklungsumgebung**

Diplomand: Daniel Merkle
Steinerweg 32
78239 Rielasingen-Worblingen

1. Prüfer: Prof. Dr. Heiko von Drachenfels (FH)
2. Prüfer: Diplom-Informatiker Rainer Börsig (FH)
Fischer Computertechnik GmbH,
Hauptstrasse 30, Stahringen
78315 Radolfzell

Ausgabedatum: 16.01.2008

Zusammenfassung (Abstract)

Thema:	Analyse, Konzeption und prototypische Umsetzung einer prozessorunabhängigen XSL-FO Entwicklungsumgebung
Diplomand:	Daniel Merkle Steinerweg 32 78239 Rielasingen-Worblingen
Firma:	Fischer Computertechnik GmbH Hauptstrasse 30, Stahringen 78315 Radolfzell
Betreuer:	Prof. Dr. Heiko von Drachenfels (Fachhochschule Konstanz)
Abgabedatum:	16.01.2008
Schlagworte:	medienneutrale Verwaltung, XML, PDF, XSL-FO, W3C Standard, Stylesheets, Seitenlayout, Seitenformatierungen, Entwicklungsumgebung, grafische Oberfläche, Projekt- Konstrukt, FO- Prozessoren, Antenna House XSL Formatter, RenderX XEP, Apache FOP, prozessorunabhängig, Vorschau

Die Firma FCT bietet industriellen Unternehmen eine Lösung im Bereich Enterprise-Content-Management und Redaktionssysteme an. Der Fokus liegt auf der medienneutralen Verwaltung der Daten im XML-Format und auf dem Publizieren dieser Daten in verschiedene Ausgabeformate wie PDF, HTML oder Online Help. Die Publikation nach PDF erfolgt meistens mit Desktop Publishing Tools wie Adobe FrameMaker, Adobe InDesign oder Microsoft Word.

Immer mehr wird die Publikation nach PDF mit dem W3C Standard XSL Formatting Objects (XSL-FO) eingesetzt. Die Publikation nach PDF erfolgt dabei vollautomatisch anhand von Stylesheets. Durch die vollautomatische Publikation hatte der Redakteur bisher keine Möglichkeit manuell Seitenlayout und Seitenformatierungen anzupassen, da das Anpassen der Stylesheets spezielles programmiertechnisches Know-How voraussetzt.

Im Rahmen dieser Diplomarbeit wurde eine Entwicklungsumgebung konzipiert und implementiert, die es Redakteuren ermöglicht, Seitenlayout und Seitenformatierung komfortabel über eine grafische Oberfläche festzulegen. Da hierbei unterschiedliche Dokumententypen und unterschiedliche Kunden berücksichtigt werden müssen, verwendet die Anwendung ein Projektansatz um die unterschiedlichen Stylesheets, Konfigurationen und Kunden zu organisieren. Da bei unterschiedlichen Kunden verschiedene FO-Prozessoren wie Antenna House XSL Formatter, RenderX XEP oder Apache FOP zum Einsatz kommen, war es ein Schwerpunkt dieser Arbeit, die Unterschiede dieser Prozessoren zu analysieren und die Entwicklungsumgebung prozessorunabhängig aufzusetzen. Eine Vorschau zeigt dem Redakteur, wie sich die verschiedenen Konfigurationen auf das PDF-Dokument auswirken, um gegebenenfalls weitere Änderungen am Seitenlayout bzw. Seitenformatierungen vorzunehmen.

1 Einleitung

Die Firma Fischer Computertechnik GmbH[1] ist spezialisiert auf die Entwicklung von XML- basierten Redaktionssystemen und Enterprise-Content-Management-Systemen. Diese Systeme erleichtern den Workflow eines technischen Redakteurs und bringen eine Unterstützung in den Bereichen Erstellung, Generierung und Übersetzung von modular aufgebauten Dokumentationen und Produktkatalogen. Hierbei kommt die Auszeichnungssprache XML, eXtensible Markup Language, zum Einsatz.

Dokumente, die von den Redakteuren erfasst werden, werden als sogenannte XML-Bausteine in einem Redaktionssystem abgelegt. Mit Unterstützung des TIM-RS Plugins, welches mit Adobe FrameMaker kompatibel ist, können Dokumente über den XML-Export in das System abgelegt und gewartet werden. Durch die Integration von XSL-Stylesheets, eXtensible Stylesheet Language, können XML-Dokumente flexibel nach HTML, Word oder PDF publiziert werden.

XSL Formatting Objects

Der Schwerpunkt dieser Diplomarbeit zielt auf das vollautomatische Erzeugen von PDF-Dateien mittels XSL-FO, vom W3C standardisiertes Seitenlayout und Umbruchsystem, ab. Das Unternehmen Fischer Computertechnik GmbH (im Folgenden mit FCT abgekürzt) nutzt für die Bearbeitung von redaktionellen Dokumenten eine flexibel anpassbare Struktur für die Auszeichnung der Inhalte. Die Elemente dieser Struktur, die durch eine DTD definiert sind, werden mit Hilfe von XSLT-Stylesheets interpretiert und nach XSL-FO umgesetzt.

XSL-FO ist die Bezeichnung für eXtensible Stylesheet Language - Formatting Objects. XSL-FO bietet sich an, wenn es darum geht, das Aussehen einer Seite zu beschreiben und zu gestalten. Diese Sprache findet ihren Einsatz in der Satz- und Umbruchgestaltung wieder. Es können mehrere Elemente einer Seite beschrieben werden, bspw. Zeichenfolgen, Grafiken, Linien und andere grafische Elemente. XSL-FO dient zur aufbereitenden Darstellung von XML-Daten. [PIKR04]

Motivation

Anspruchsvolle Satz- und Umbruchgestaltung wird von professionellen Schriftsetzern erstellt. Die Grundlage zur Generierung der PDF-Dokumente liefern XSL-FO Stylesheets. Die Anpassung dieser Stylesheets erfordert programmiertechnisches Know-How und wird bislang eher von Software-Entwicklern durchgeführt, für die der Automatisierungsaspekt im Vordergrund steht. Der Aspekt der Satz- und Umbruchgestaltung ist für sie eher nachrangig.

Redakteure hingegen haben die Erfahrung in den Bereichen Satz, Layout und Typografie, jedoch im Normalfall keine Kenntnisse in der Programmierung oder XSL-Entwicklung. Der Redakteur war daher bislang nicht in der Lage XSL-Stylesheets anzupassen oder gar neue Stylesheets zu entwickeln. Erfahrene Programmierer werden keine Schwierigkeiten besitzen, sich in die Implementierung eines XSL-Stylesheets hineinzuversetzten.

Im Rahmen dieser Diplomarbeit wird eine Entwicklungsumgebung konzipiert und implementiert, die es einem Redakteur ermöglicht, Seitenlayout und Seitenformatierung komfortabel über eine grafische Oberfläche festzulegen, ohne dass er dazu spezielle Programmierkenntnisse benötigt.

[1] http://www.fct.de

Praktische Herangehensweise

Die Kunden von FCT besitzen unterschiedliche FO-Prozessoren, die die Ausgabe des PDF-Dokuments ermöglichen. Dieser Umstand machte eine Analyse der möglichen FO-Prozessoren Antenna House XSL Formatter, RenderX XEP und Apache FOP nötig. Hierbei wurden die Unterstützungen des FO-Standards und die Unterschiede in den grafischen Darstellungen zusammen mit den Gemeinsamkeiten genaustens analysiert (siehe Kapitel 4, Seite 13).

Nach dieser Analyse wurden die Anforderungen an die XSL-FO Entwicklungsumgebung festgelegt (siehe Kapitel 5, Seite 42). Dieser Teil der Diplomarbeit beschreibt die funktionellen Anforderungen der Anwendung gegenüber dem Anwender genaustens beschrieben.

Nachdem die Anforderungen dokumentiert worden sind, wurde das Konzept der prozessorunabhängigen XSL-FO Entwicklungsumgebung erarbeitet. Zudem wurde ein Prototyp implementiert, der dem Anwender eine komfortable grafische Oberfläche zur Anpassung und Erstellung des Seitenlayouts und Seitenformatierungen bereitstellt (siehe Kapitel 6, Seite 64).

Um den Umgang mit der Anwendung und vor allem die erzielten Erfolge gegenüber den Anforderungen gewährleisten zu können, wurde die Anwendung mit verschiedenen Seitenlayouts und Seitenformatierungen aus unterschiedlichen Kunden-Dokumentationen getestet (siehe Kapitel 7, Seite 93).

Neben dem Projektablauf wurden die erzielten Ergebnisse dieser Diplomarbeit dokumentiert.

2 Das Satz- und Umbruchsystem von Formatting Objects

XSL-FO steht für eXtensible Stylesheet Language Formatting Objects, ein vom W3C standardisiertes Seitenlayout und Umbruchsystem. Dabei handelt es sich nicht um eine Sprache zur Beschreibung von verschiedenen Seiten. Vielmehr können unterschiedliche Regeln eines Seitenlayouts, wie das Auftreten eines Seitenendes und Seitenformatierungen, z.B. Fußnoten am unteren Rand einer Seite, spezifiziert werden. Jedoch wird nicht festgelegt, wie die Elemente auch tatsächlich auf den einzelnen Seiten platziert werden. Dies wird von einem sogenannten Formatierer festgelegt. [@ADOB]

Das Ergebnisdokument, das durch ein XSL-FO Stylesheet erzeugt wird, ist eine feste Folge von Seiten mit einem spezifizierten Seitenformat. XSL-FO definiert dabei die Seitenaufteilung (dies betrifft die Kopf- und Fußzeile, linke und rechte Marginalie sowie dem eigentlichen Inhalt einer Seite), Zeilen- und Seitenumbrüche, usw. Somit ist es nun möglich, über XSL-FO zu bestimmen, wo einzelne Elemente auf einer Seite angeordnet und spezifiziert werden sollen.

Angewendet wird diese Technologie vor allem in den Bereichen juristischer Publikationen und Technischer Dokumentation. Dabei handelt es sich um die Verarbeitung von Dokumenten mit großem Umfang. Aus einem generierten FO-Dokument können nun über bestimmte FO-Prozessoren PDF-Dokumente und andere Print-Medien erzeugt werden. [PIKR04]

2.1 Technischer Aspekt von Formatting Objects

Als Grundlage für Seitenformatierungen in Dokumentenverarbeituns- oder Satzsystemen dienen Seitenvorlagen. Eine Seitenvorlage dient zum Gliedern der einzelnen Seiten und legt das eigentliche Seitenlayout fest. Die kommende Abbildung zeigt die verschiedenen Möglichkeiten, eine Seitenvorlage in ihre Seitenmaße zu unterteilen.

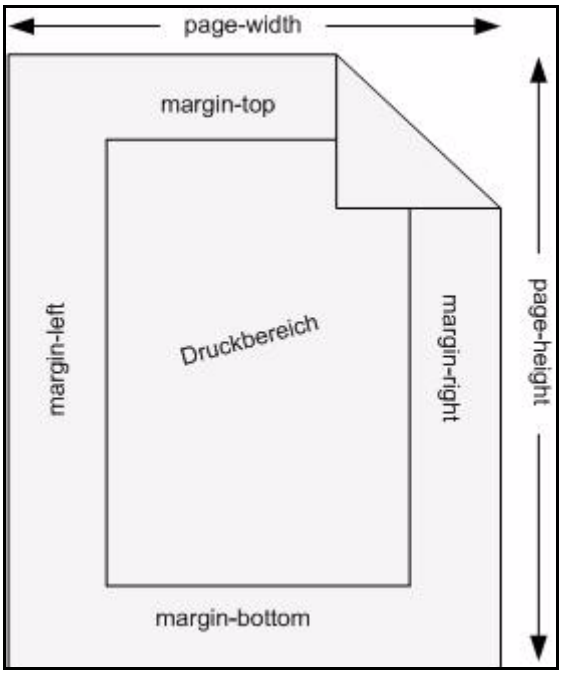


Fig. 2-1 : Der Aufbau einer Seite in XSL-FO nach [PIKR04]

Zur Definition des Seitenlayouts bietet XSL-FO die Möglichkeit, verschiedene Einstellungen betreffend einer Seite anzupassen. Zum einen wird eine Seite in ihrer Höhe und Breite über die Attribute `page-width` und `page-height` festgelegt. Optional zum Seitenformat kann eine Seite in vier Bereiche unterteilt wer-

den. Diese betreffen den Abstand des Randes nach oben (`margin-top`), nach unten (`margin-bottom`), nach links (`margin-left`) und nach rechts (`margin-right`).

Der Druckbereich einer Seite lässt sich nun in weitere fünf unterschiedliche Bereiche unterteilen. Diese Bereiche betreffen den Kopfbereich, den Fußbereich, den Hauptbereich, in dem der Seiteninhalt dargestellt wird, sowie den linken und rechten Bereich. Die folgende Abbildung zeigt diese Bereiche nochmals deutlich auf.

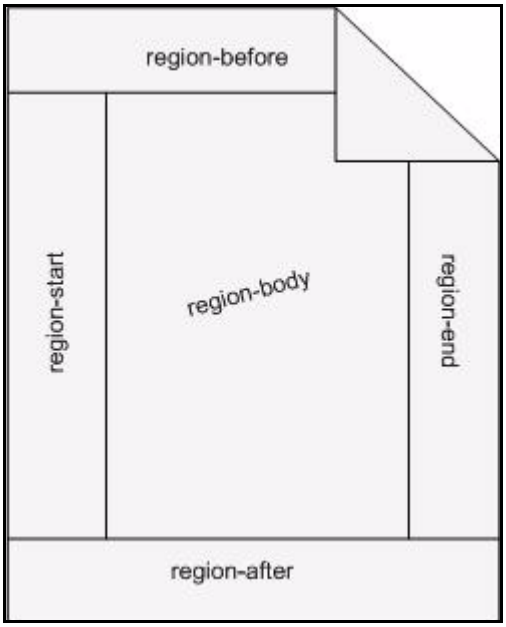


Fig. 2-2 : Fünf Bereiche eines Druckbereichs nach [PIKR04]

Der Kopfbereich wird durch das Element `region-before`, die Fußzeile durch `region-after`, der Hauptbereich durch `region-body` und der linke und rechte Bereich jeweils durch `region-start` und `region-end` beschrieben. Somit lassen sich die äußeren Ränder des Hauptbereichs, Mehrspaltigkeit im Hauptbereich sowie typografische Details für die genannten fünf Seitenbereiche festlegen. [KRUE06]

XSL-FO bietet die Möglichkeit, eine Sequenz von Seitenvorlagen zu definieren. Dies bietet sich an, wenn sich die einzelnen Seiten in Form von geraden oder ungeraden Seitenmerkmalen unterscheiden. Die folgende Abbildung zeigt ein solches Beispiel. Sequenzen von Seitenvorlagen können zusammengefasst werden um zu bestimmen, welche Vorlagen sich auf die geraden und welche Vorlagen sich auf die ungeraden Seiten beziehen. Eine Seite spezifiziert sich von außen nach innen. Dieses Prinzip wird auch "*Von-außen-nach-innen*" genannt und setzt sich bei untergeordneten Objekten wie Blöcken oder Tabellen fort. Dies hat den Vorteil, dass sich verschiedenen Objekte einfach verschachteln lassen. [KRUE06]

Für Bücher oder technische Handbücher werden Seitenfolgenvorlagen verwendet, um rechte und linke Seiten eindeutig spezifizieren zu können. In der Abbildung Fig. 2-3 kann man deutlich den Umbruch von gerader auf ungerader Seite erkennen. Dieser Umbruch kommt zustande, wenn der Textfluss über die definierten Randeinstellungen hinausgeht, bzw. die Seite gebrochen werden muss. Wenn dies zutrifft, dann folgt der Textfluss auf einer neuen Seite.

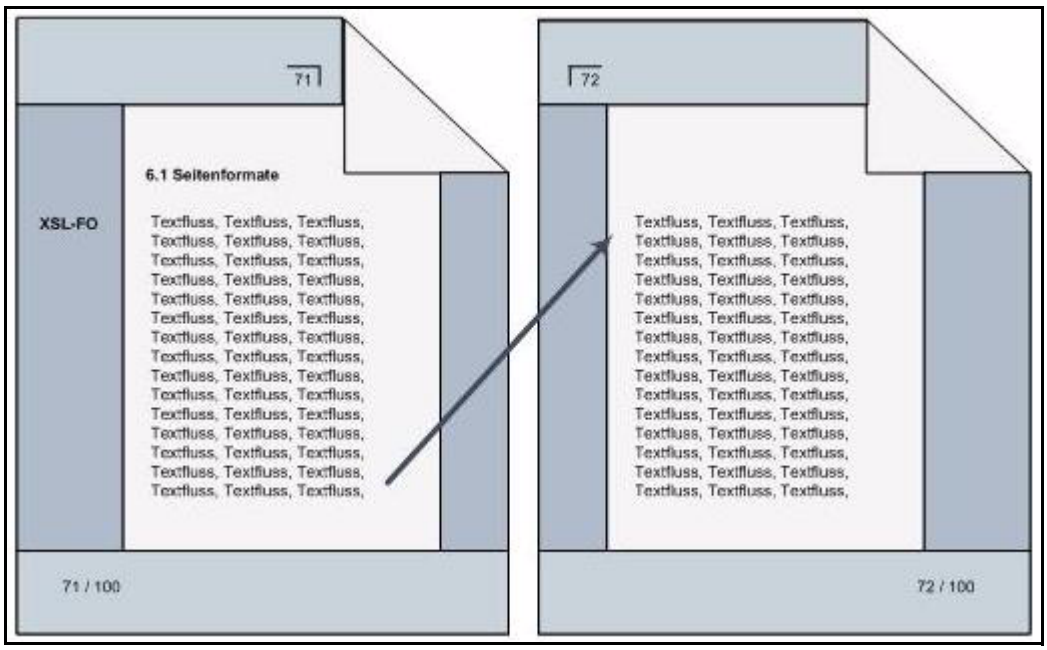


Fig. 2-3 : Seitenumbruch bei geraden und ungeraden Seiten

Die so entstehenden Seitenfolgen referenzieren die entsprechenden Seitenvorlagen und wissen dadurch, welche Bereiche und welche Randeinstellungen bzw. Seitenformate angewendet werden sollen. Diese Bereiche werden in XSL-FO durch Absatzformate, Gestaltungsmerkmale sowie Positionierung innerhalb einer Seite definiert.

2.2 Qualitätsaspekt von Formatting Objects

XSL-FO bietet den Aspekt einen qualitativ hochwertigen Satz zu schaffen. Dies bedarf jedoch eines gewissen programmiertechnischen Know-Hows. Ein Schriftsetzer, der sich mit einem hochwertigen Satz auskennt, hat jedoch keine programmiertechnischen Kenntnisse in XSL-FO. Auf der anderen Seite gibt es den Software-Entwickler, der sich zwar mit XSL-FO auskennt, jedoch nicht mit einem qualitativ hochwertigen Satz. Diese Herausforderung soll mit dieser Diplomarbeit betrachtet werden. Es soll eine Anwendung konzipiert und implementiert werden, die es einem Redakteur ermöglicht, sein Know-How im Bereich Seitenlayout und Seitenformatierung einzusetzen ohne dabei XSL-FO anwenden zu müssen.

3 Die Firma Fischer Computertechnik GmbH

Die Firma FCT, gegründet 1985 in Frankfurt/Main, ist eines der führenden Entwickler und Anbieter von Redaktions- und Enterprise Content-Management-Systemen, basierend auf XML. Dabei unterstützt das Unternehmen technische Redakteure und Marketing-Abteilungen bezüglich automatisiertem Erstellen, Pflegen, Aktualisieren, Übersetzen und Publizieren von Technischen Dokumentationen und Produktkatalogen. [@FCT]

3.1 Erstellung von qualitativ hochwertigen Dokumentationen

Industrielle Untenehmen erfassen Technische Dokumentationen (Text, Bild) in einem Editor, z.B. Adobe FrameMaker. Anschließend werden Sinneinheiten gespeichert und als XML-Bausteine im Redaktionssystem angelegt. Die folgende Abbildung verdeutlicht den Begriff Bausteine.

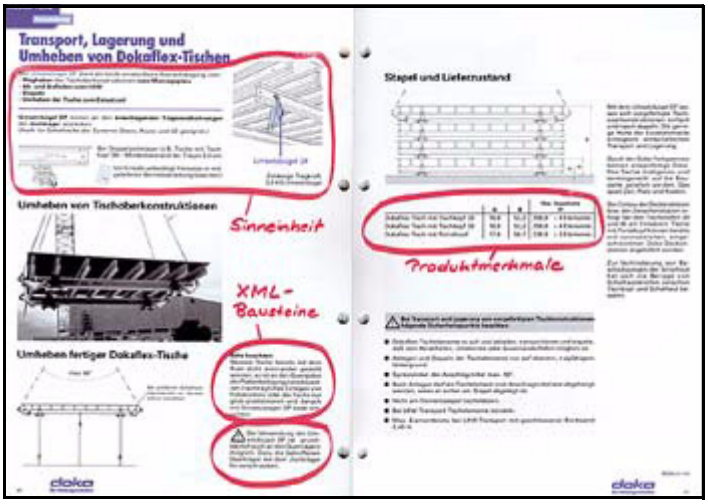


Fig. 3-1 : Technische Dokumentation anhand [@FCT]

Bausteine können bestimmte Absätze oder Kapitel einer Dokumentation sein. Sie zeichnen sich dadurch aus, dass sie möglichst oft in anderen Dokumenten wiederverwendet werden können. Dazu wird ein Baustein aus dem Redaktionssystem an der gewünschten Stelle im Dokument eingefügt. Sollte sich der Baustein inhaltlich verändern, muss er nur in einer Verwendung bearbeitet werden. Die übrigen Verwendungen können einfach aktualisiert werden.

Über Adobe's FrameMaker kann der Kunde großen Einfluss auf die Gestaltung des Absatzes und das Seitenlayout ausüben. Aus FrameMaker heraus kann das Dokument nach PDF gespeichert werden. Aber auch Kunden, die keinen Editor einsetzen, können durch den Export der Bausteine bzw. einer vollständigen Dokumentation durch bestimmte Prozesse diese strukturierten Inhalte in verschiedene Formate publizieren, z.B. PDF. Hierfür gibt es erste Umsetzungen nach FO durch XSL-FO Stylesheets.

3.2 Qualitativ hochwertiger Satz vs. XSL-FO Entwicklung

XSL-FO Stylesheets werden von Software-Entwicklern implementiert. Die Anforderungen an diese Stylesheets kommen von unterschiedlichen Redakteuren. Sie geben den Software-Entwicklern einen einmaligen Überblick über die grafische Gestaltung und Seitenformatierung der zu erzeugenden PDF-Dokumente. Auf weitere Änderungen, die nach der Entwicklung des XSL-FO Stylesheets eingehen würden, hat der Redakteur keinen weiteren Einfluss.

Die verschiedenen Kunden von FCT haben unterschiedliche Anwendungen im Einsatz, um Technische Dokumentationen zu erstellen. Über Adobes FrameMaker können Vorlageseiten und Absatzformate definiert und für den Inhalt der Seiten verwendet werden. Dabei hat nicht jeder Redakteur das Recht, Seitenvorlagen zu pflegen oder zu erstellen. Durch diese Diplomarbeit soll dem Redakteur die Möglichkeit eingeräumt werden, nun selbst das Seitenlayout, Seitenformatierungen und Absatzformate bestimmen zu können, ohne dass er Einfluss auf das entwickelte XSL-FO Stylesheet nehmen muss.

4 Analyse der FO-Prozessoren

In diesem Abschnitt der Diplomarbeit werden die FO-Prozessoren Antenna House XSL Formatter V4.2, Apache FOP V0.20.5 und RenderX XEP analysiert betreffend ihrer Gemeinsamkeiten, ihrer Unterschiede sowie Problemen bei der Formatierung von FO-Dokumenten. Diese Analyse soll Aufschluss darüber geben, welche Attributs- und Elementdefinitionen die FO-Prozessoren gemeinsam haben, um daraus für die prozessorunabhängige XSL-FO Entwicklungsumgebung eine Grundlage zu schaffen. Diese Diplomarbeit beschränkt sich auf diese Prozessoren, da es die gängigsten sind, die vom Unternehmen FCT eingesetzt werden. Bevor die einzelnen Prozessoren näher analysiert werden, wird der Verarbeitungsschritt von XSL-FO Dokumenten erläutert, um zu verstehen, wie die einzelnen Prozessoren das XML-FO Dokument in die jeweilige Ausgabe-Datei übertragen.

4.1 Der Verarbeitungsprozess eines FO-Prozessors

Bei der Verarbeitung eines XSL-FO Dokuments bis zur Überführung in das Ausgabeformat, können zwei unterschiedliche Verarbeitungsprozesse angewendet werden. Die folgende Abbildung zeigt die erste Möglichkeit auf.

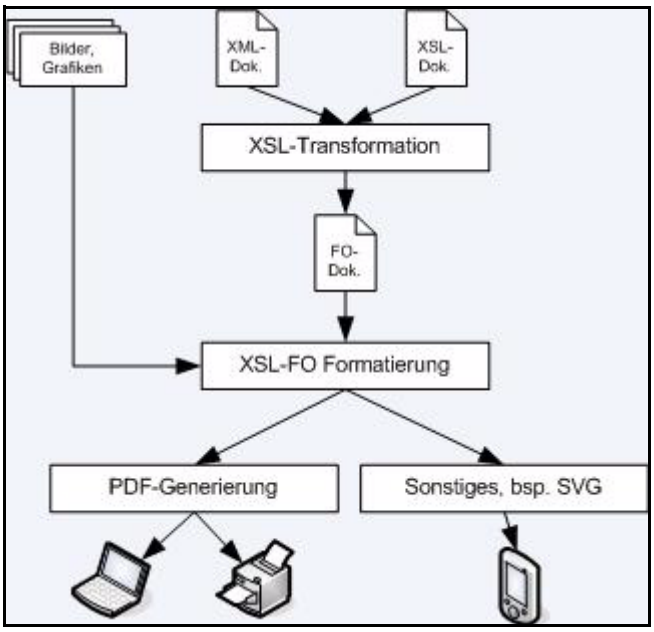


Fig. 4-1 : XSL-FO Verarbeitungsprozess nach [KRUE06]

In diesem Szenario spielen einige Objekte eine große Rolle. Zum einen gibt es das XML-Dokument, welches die Struktur des zu verarbeitenden Dokuments aufweist. In diesem XML-Dokument befinden sich XML-Bausteine, die Inhalte bezüglich Technischen Dokumentationen besitzen. Diese Inhalte sind durch Elemente ausgezeichnet, um anhand dieser ihre Semantik wiederzugeben. Dieses XML-Dokument wird über ein XSL-Stylesheet geparst und verarbeitet. Das Stylesheet hat demnach gewisse Regeln, welche ein gültiges FO-Dokument erzeugen. Diese Regeln interpretieren die jeweiligen Elemente des XML-Dokuments und setzen dafür neue Elemente, bspw. Formatting Objects, und fügen diesen die Inhalte aus dem XML-Dokument bei. Ein XSLT-Prozessor, bspw. Xalan[1], erzeugt aus diesen Eingabedokumenten mithilfe der XSL-Stylesheets das Ergebnisdokument. Wie in der Abbildung ersichtlich handelt es sich hierbei um das XML-FO-Dokument.

[1]zur freien Verfügung unter http://xml.apache.org/xalan-j/.

Im nächsten Schritt wird das Ergebnisdokument an einen FO-Prozessor übergeben. Dieser FO-Prozessor interpretiert und formatiert die jeweiligen Objekte des FO-Dokuments und erzeugt die passende Ausgabe, z.B. PDF bzw. SVG-Dateien. Eine andere Vorgehensweise zeigt die nachfolgende Abbildung (Fig. 4-2).

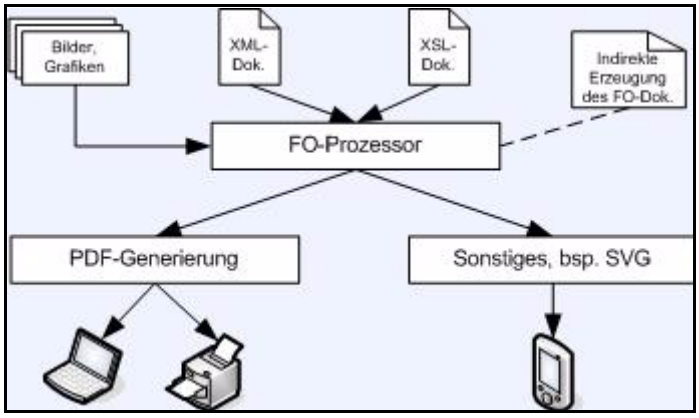


Fig. 4-2 : XSL-FO Verarbeitungsprozess mittels FO-Prozessor

Wie im ersten Szenario gibt es hier das XSL-Stylesheet, welches aus dem XML-Dokument ein FO-Dokument erzeugt. Diese beiden Dokumente werden nun nicht an einen XSLT-Prozessor, sondern direkt an einen XSL-Formatierer übergeben. Das hat den Vorteil, dass der Formatierer den XSLT-Prozessor selbst im Hintergrund ausführt, welcher als Ergebnisdatei das FO-Dokument übermittelt [KRUE06].

Dieses Dokument kann nun weiterverarbeitet werden, um daraus eine PDF-Datei zu erstellen. Wenn dieser Prozess über die Kommandozeile aufgerufen wird, können Fehler durch den XSLT-Prozessor abgefangen und ausgegeben werden. Diese beziehen sich auf bestimmte Zeilen und Spalten der XML-Datei bzw. der XSL-Datei und können so schnell gefunden und behoben werden.

4.2 FO-Prozessoren im Überblick

Im Folgenden werden nun die einzelnen FO-Prozessoren Apache FOP, Antenna House XSL Formatter und RenderX XEP vorgestellt, um einen Eindruck über die Eigenschaften der einzelnen Prozessoren zu vermitteln. Im weiteren Verlauf wird näher darauf eingegangen, welche Aspekte des FO-Standards von den untersuchten Prozessoren unterstützt werden.

4.2.1 Apache FOP

Die Abkürzung FOP steht für Formatting Objects Processor. Dieser FO-Prozessor steht im Internet zur freien Verfügung[1]. Es handelt sich hierbei um eine in Java verfasste Anwendung, die ein Formatting Object-Dokument, im Folgenden durch FO-Dokument abgekürzt, einliest und in ein vom Anwender gewähltes Ausgabeformat überträgt. Dabei unterstützt FOP die Ausgabeformate PDF, PCL[2], PostScript, SVG[3], XML, Print, AWT, MIF und TXT, wobei PDF das Standardausgabeformat ist. [@APACb] In der weiteren Analyse wird detailliert auf Vor- und Nachteile des FOP eingegangen.

4.2.2 Antenna House XSL Formatter

Der Antenna House XSL Formatter[4] bietet Unterstützung für über fünfzig Sprachen, PDF Generierungen sowie eine grafische Benutzeroberfläche. Dieser FO-Prozessor unterstützt Formate wie SVG, EMF, WMF, MathML, WordML und PostScript. Das Ansprechen kann über die Kommandozeile oder über die Integration in eine Anwendung erfolgen. Der XSL-Formatter bringt ein enormes Leistungsspektrum mit sich bezüglich Anpassung und Positionierung der einzelnen Objekte auf einzelnen Seiten, Silbentrennung gegenüber internationalen Sprachen und Zeichensätzen, Schreibrichtungen, Gestaltung von komplexen Konstrukten im Bereich Tabellengestaltung, Grafikunterstützung sowie Integration bzw. Einbettung anderer PDF-Dokumente. Einen umfangreicheren Überblick bekommt man auf der Internetseite von Antenna House. Der Formatter steht leider nicht zur freien Verfügung, kann jedoch als Trial Version für dreißig Tage getestet werden. [@MIDH]

4.2.3 RenderX XEP

XEP ist sowohl ein XSL-FO Prozessor als auch ein Prozessor für Scalable Vector Graphics, bestehend aus einer Ansammlung von Java Klassen. Über die Kommandozeile können ihm als Eingabedokumente XSL-FO und XML-Dokumente übergeben werden. Das XSL-FO Stylesheet enthält das Layout und den Aufbau der Seiten, welche die unterschiedlichen Elemente des XML-Dokuments darstellen. Dieses Stylesheet wird eingesetzt, um das eigentliche FO-Dokument zu erzeugen. Der Vorgang wird indirekt über den Prozessor gesteuert und überträgt das Ergebnisdokument nach PDF, PostScript oder AFP. Wie der Formatter von Antenna House ist auch diese FO-Formatierer nicht frei erhältlich. Er kann jedoch ebenso als Trial Version für dreißig Tage getestet werden. [@REND]

[1] http://xmlgraphics.apache.org/fop/download.html

[2] steht für Printer Command Language, eine Programmiersprache mittels derer Befehle an HP Deskjet-Drucker übertragen werden können. [@HP]

[3] steht für Scalable Vector Graphics, durch diese Sprache können 2-dimensionale Grafiken in XML dargestellt werden. [@SVG]

[4] http://www.antennahouse.com/product/axfo40/axfo4top.htm

4.3 Unterstützung des FO-Standards

Bevor auf die Unterschiede der einzelnen FO-Prozessoren eingegangen wird, lenkt dieses Kapitel das Augenmerk auf die unterschiedlichen Probleme, die manche FO-Prozessoren mit sich bringen. Dabei wird auf diejenigen Elemente und Attribute eingegangen, die durch den Prozessor bei der Überführung in das Ergebnisdokument in den vorliegenden Versionen noch nicht unterstützt werden. Dieses Kapitel soll eine Hilfestellung dafür sein, wie die prozessorunabhängige Anwendung aufgebaut werden soll.

4.3.1 Der Seitenaufbau

Eine Seite lässt sich definieren in den Eigenschaften Höhe und Breite, sowie Rändermaße wie oberer, unterer, linker und rechter Abstand. In dem Element `fo:simple-page-master` können diese Angaben definiert werden, wobei diese durch die Attribute `page-height`, `page-width`, `margin-left`, `margin-right`, `margin-top` und `margin-bottom` dargestellt werden.

Hierbei gibt es keine wesentlichen Unterschiede in den Darstellungen der einzelnen FO-Prozessoren. Jedoch treten im Bereich der Seiteninhalte (Druckbereich eines Dokuments) einige Problematiken auf, die im folgenden Kapitel näher analysiert werden.

Der Druckbereich

Der Inhalt einer Seite, ohne Ränder, kann in die folgenden fünf Kategorien eingeteilt werden: Hauptbereich, linker und rechter Seitenbereich, Fußbereich sowie Kopfbereich. Dabei stellt das Element `fo:region-body` den Hauptbereich, das Element `fo:region-start` den linken Bereich, das Element `fo:region-end` den rechten Bereich, das Element `fo:region-before` den Kopfbereich und das Element `fo:region-after` den Fußbereich dar.

[PIKR04]

Nun kann man bestimmte Einstellungen über die Attribute dieser Elemente steuern. Es können Einstellungen bezüglich Grafiken über die Attribute `background-image` und `background-repeat`, Ränder über `border-after-color`, `border-after-style`, `border-after-width`, `border-before-color`, `border-before-style`, `border-before-width`, `border-top-color`, `border-top-style`, `border-top-width`, `border-bottom-color`, `border-bottom-style` und `border-bottom width` oder Schreibrichtungen und Ausrichtungen über `reference-orientation` und `writing-mode` getätigt werden.

Manche FO-Prozessoren haben ihre Schwierigkeiten damit, diese Attribute zu unterstützen. Man wird gleich sehen, welche Prozessoren die hier genannten Attribute umsetzen können und welche noch keine Unterstützung dafür anbieten.

Apache FOP

Bislang ist es so, dass der FO-Prozessor von Apache noch keine Grafiken und Ränder bei den Elementen `fo:region-<Bereich>` unterstützt. Es können zwar schon Grafiken angezeigt werden, jedoch lassen sich diese nicht beliebig verschieben, d.h. die Positionierung und die Veränderung von Höhe und Breite lassen sich nicht beeinflussen. Werden die entsprechenden Attribute dennoch in das FO-Dokument mit integriert, so wird beim Verarbeitungsprozess über die Konsole eine dementspreche Warnung ausgegeben. Die Transformation findet trotzdem statt, nur ohne Formatierung der entsprechenden Objekte, z.B. werden Grafiken dann im PDF-Dokument standardmäßig linksbündig und in Originalgröße angezeigt. Bei der Angabe von Linien (Rändern) geschieht im Ergebnisdokument keine Veränderung. Sie können erst gar nicht dargestellt werden.

Es gibt jedoch eine andere Möglichkeit, Grafiken und Ränder mit in das Dokument aufzunehmen. Die Einstellungen finden im Element `fo:page-sequence` statt. Dieses Element gibt eine Seitenfolge mit Inhalten wieder. Hier gibt es das Element `fo:static-content`, in dem Kopf- und Fußzeile, sowie linker und rechter Seitenbereich mit Inhalten gefüllt werden können. Die Inhalte bestehen aus `fo:block` Elementen. Hier kann nun durch die Verwendung einer Tabelle über das Element `fo:table` eine Linie, beispielsweise in den Kopfbereich integriert werden. Hierzu gibt es Attribute wie `border-after-style` und `border-after-color`. Die folgende Abbildung zeigt, wie dies erreicht werden kann.

```
<fo:static-content>
    <fo:table>
        <fo:table-body border-top-style="solid"
                       border-top-color="black"
                       border-top-width="1pt">
            <fo:table-row>...</fo:table-row>
        </fo:table-body>
    </fo:table>
</fo:static-content>
```

Fig. 4-3 : Liniendarstellung mit Apache FOP

Um eine Grafik in einen Bereich einzufügen wird das Element `fo:external-graphic` in den Block eingefügt. In den weiteren Kapiteln wird dies deutlicher.

Ein weiteres Problem bei Grafiken ist, dass nicht jedes Grafikformat von FOP unterstützt wird. Formate wie *.gif* und *.eps* werden nicht umgesetzt und können so auch nicht dargestellt werden. Im Ergebnisdokument selbst würde die Grafik fehlen bzw. wäre nicht sichtbar, obwohl sie im FO-Dokument vorhanden ist.

Wenn die Grafik über das Attribut `background-position-horizontal` verschoben werden soll, so ist auch hier das Problem, dass egal welchen Wert dieses Attribut annimmt, keine Änderung im Ergebnisdokument zu finden ist.

Ein weiteres Problem spielt sich beim Attribut `reference-orientation` ab. Der Attributwert sollte den Block um die angegebene Gradzahl neigen. FOP ignoriert das Attribut vollständig. Die Standardausgabe ist linksbündig und von oben nach unten.

Wenn man sich die Weltsprachen etwas näher anschaut, so gibt es doch einige Unterschiede, was die Schreibrichtung der einzelnen Texte angeht. Um die Schreibrichtung in XSL-FO zu bestimmen, wird das Attribut `writing-mode` verwendet. Dies kann sowohl in den Elementen `fo:simple-page-master, fo:table` als auch für `fo:region-<Bereich>` verwendet werden. Hierzu können drei Werte angenommen werden: `rl-tb, lr-tb` und `tb-rl`. [@W3SC] Je nach Einsatz erstreckt sich die Schreibrichtung von rechts nach links, von links nach rechts wobei hier von oben nach unten geschrieben wird und von oben nach un-

ten in der Richtung rechts nach links. Die nachfolgende Abbildung zeigt nochmals die Änderungen des Attributwerts.

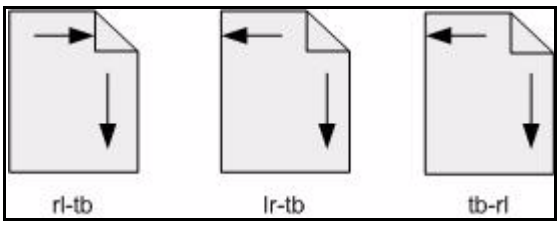


Fig. 4-4 : Schreibrichtungen in XSL-FO nach [PIKR04]

Beim Überführen des FO-Dokuments in das Ergebnisdokument werden diese Einstellungen mit dem Apache FOP nicht berücksichtigt. Es ist keine sichtbare Veränderung vorhanden. Standardmäßig ist hier die Schreibrichtung *rl-tb* gesetzt worden.

Antenna House XSL Formatter

Der FO-Prozessor Formatter von Antenna House unterstützt bei der Interpretation des Seitenlayouts bis auf wenige Ausnahmen jedes Attribut. Somit ist es möglich Angaben bezüglich Kopf- und Fußzeile, linkem und rechtem Seitenrand betreffend Grafiken und Rändern zu treffen, die dann nach der Überführung in das Ergebnisdokument nach Interpretation des Prozessors dargestellt werden.

Jedoch hat der Formatter ebenso wie FOP ein Problem bei der Unterstützung des Attributs `reference-orientation`. Hier erfolgt ebenso keine sichtbare Änderung im Ergebnisdokument. Die Standardausgabe ist linksbündig und von oben nach unten.

Ebenso ist es nicht möglich, die Schreibrichtungen über das Attribut `writing-mode` konsequent zu steuern. Im Element `fo:table` wird die Schreibrichtung richtig umgesetzt. Keine Unterstützung gibt es jedoch für die Elemente `fo:simple-page-master` und `fo:region-<Bereich>`.

RenderX XEP

Dieser FO-Prozessor unterstützt die meisten Attribut, auch `reference-orientation`. Jedoch hat auch dieser Prozessor Probleme, was die Schreibrichtung in Bezug auf das Element `fo:table` anbelangt. Wenn das Attribut `writing-mode` den Wert *tb-rl* annimmt (top to bottom, right to left), so gibt es im Ergebnisdokument keine sichtbaren Änderungen. Alle anderen Werte werden richtig umgesetzt.

4.3.2 Farbdeklarationen

In XSL-FO können Farbprofile definiert werden. Mit dem Element `fo:declaration` ist es möglich, über Kinderelemente der Struktur `fo:color-profile` bestimmte Farbprofile zu deklarieren und zu referenzieren. Somit ist die Möglichkeit gegeben, eigene Farbprofile zu erstellen, die dann im FO-Dokument bspw. als Hintergrundfarben verwendet werden können.

Das Farbprofil

Das Element `fo:color-profile` besitzt die beiden Attribute `src` und `color-profile-name`. Über die Eigenschaft `src` kann eine Referenz auf ein extern vorhandenes Farbprofil hinterlegt werden. Diese kann dann im FO-Dokument über das `color-profile-name` referenziert werden. Die untenstehende Abbildung verdeutlicht ein Beispiel einer solchen Deklaration.

```
<fo:declarations>
    <fo:color-profile color-profile-name="colorProfile"
                      src="color.icc">
</fo:declarations>
```

Fig. 4-5 : Farbprofil mit XSL-FO

Hier wurde ein Farbprofil mit dem Namen *colorProfile* angelegt. Das Farbprofil bezieht sich auf die lokal liegende Datei ***color.icc***. Nach dieser Deklaration kann die Farbe nun im gesamten FO-Dokument eingesetzt werden. Dies geschieht bspw. im Block-Element über das Attribut `color`. Die Verwendung sieht folgendermaßen aus.

```
<fo:block color="rgb-icc(255,0,0,#colorProfile,1,0,0">
    .....
</fo:block>
```

Fig. 4-6 : Referenzierung eines Farbprofils

Apache FOP

Bei der Verwendung eines Farbprofils kommt es bei Apache FOP zu Problemen. Die Farbdeklaratioin wird vollständig ignoriert. In diesem Beispiel wäre die Farbe des Textes standardmäßig schwarz.

Die einzige Möglichkeit der Umsetzung besteht darin, eine feste Integration der jeweiligen Farbe in die Attribute `color` bzw. `background-color` des Block Elements mit aufzunehmen.

Antenna House XSL Formatter

Der Formatierer von Antenna House hat diese Unterstützung mit aufgenommen und kann somit Farbprofile im gesamten FO-Dokument referenzieren und deklarieren.

RenderX XEP

RenderX XEP unterstützt genauso wie Antenna House XSL Formatter die Definition eines Farbprofils.

4.3.3 Seiteninhalte

Das Element `fo:page-sequence` ist zuständig für die Inhalte einer Seite. Es hat die folgenden Kindelemente: `fo:title`, `fo:static-content` und `fo:flow`. Die Inhalte zu Kopf- und Fußzeile, sowie linkem und rechtem Seitenbereich können mit dem Element `fo:static-content` umgesetzt werden. Die Inhalte, die sich mitten in der Seite befinden bzw. Inhalte wo es Seitenumbrüche und die damit verbundene Anzahl von dynamisch erzeugten Seiten gibt, werden über das Element `fo:flow` gesteuert. [PIKR04]

Im folgenden Teil dieser Diplomarbeit wird nun näher auf die Probleme mit den einzelnen Inhalten eingegangen, die sich in den Block-Elementen `fo:block` befinden.

Kapitelüberschriften in der Kopfzeile

Mit dem Element `fo:retrieve-marker` kann die Überschrift des aktuellen Kapitels in der Kopfzeile wiedergegeben werden. Dies kommt in den häufigsten PDF-Dokumenten vor und erleichtert den Überblick über das aktuelle Kapitel. Durch das Attribut `retrieve-marker-class-name` wird die Überschrift in die Kopfzeile kopiert. Hier gibt es auch einige Probleme, die durch die FO-Prozessoren entstehen. Zuvor ein kleines Beispiel zur Veranschaulichung.

```
<fo:retrieve-marker retrieve-boundary="page"
                    retrieve-position="first-including-carryover"
                    retrieve-class-name="H1_Heading"/>
```

Fig. 4-7 : Kapitelüberschriften in Kopfzeilen

Apache FOP

Die Eigenschaften `retrieve-boundary` und `retrieve-position` dürfen keine leeren Werte besitzen. Es wird eine dementsprechende Fehlermeldung über die Konsole ausgegeben. Das Ergebnisdokument wird nicht erzeugt.

Ebenso gibt es Probleme, wenn um das Marker-Element ein Block-Element gesetzt wird. Wenn die Schriftart, Schriftgröße und der Schriftstil dabei angegeben werden, so werden diese in Bezug auf die Kapitelüberschrift nicht berücksichtigt.

Antenna House XSL Formatter

Der Antenna House XSL Formatter wird allen Anforderungen gerecht. Kapitelüberschriften in der Kopfzeile können problemlos formatiert und referenziert werden.

RenderX XEP

Auch der FO-Prozessor XEP kann nicht mit leeren Attributwerten umgehen. Wenn diese nicht vorhanden sind, wird das Ergebnisdokument nicht erzeugt. Schrifteinstellungen, die sich auf den Marker auswirken, werden von XEP unterstützt.

Tabellenstrukturen

Mit XSL-FO ist es möglich, komplexe Tabellenstrukturen mittels `fo:table` umzusetzen. Diese Struktur ist recht einfach gehalten. Um einzelne Spalten definieren zu können, wird das Element `fo:table-column` verwendet. Um jeweils eine Tabellenkopf- bzw. Tabellenfußzeile zu erzeugen, bedarf es der Elemente `fo:table-header` und `fo:table-footer`. Der eigentliche Inhalt der Tabellen in den jeweiligen Reihen und Spalten wird im Element `fo:table-body` abgebildet.

Tabellen haben auch die Möglichkeit, Ränder an den Seiten zu zeichnen. Dazu dient das Attribut `border-<Bereich>-style` um den Stil einer Linie zu bestimmen. Dabei kann dieses Attribut die Werte *solid*, *none*, *hidden*, *dotted*, *dashed*, *double*, *groove*, *ridge*, *inset* und *outset* annehmen. [PIKR04]

Apache FOP

Das Attribut `table-layout` muss bei diesem Element mit angegeben werden. Wird es nicht mit angegeben, wird eine dementsprechende Warnung des FO-Prozessors über die Konsole mit ausgegeben. Die Eigenschaft `table-layout="auto"` wird nicht von Apache FOP unterstützt. Stattdessen wird der Wert des Attributs auf `fixed` gesetzt.

Ein weiteres Problem, das sich auf FOP bezieht, ist die Linienausprägung am Beispiel der Tabellenränder. Die einzigen Werte, die bei den Angaben des Linienstils umgesetzt werden können, sind *solid*, *none* und *hidden*. Bei allen anderen Werten gibt es keine sichtbaren Änderungen im Ergebnisdokument.

Antenna House XSL Formatter

Die Angabe des Attributs `table-layout` ist nicht zwingend notwendig. Die Überführung des FO-Dokuments in das Ergebnisdokument verläuft fehlerfrei. Jedoch ist es immer von Vorteil, wenn diese Angabe mit angegeben wird.

Wenn Ränder der Tabelle hervorgehoben werden sollen, so hat der Formatter hier keine Probleme. Er unterstützt jegliche Art von Linien.

RenderX XEP

Ebenso wie beim Antenna House XSL Formatter, ist die Angabe des Attributs `table-layout` auch hier nicht zwingend notwendig. Der FO-Prozessor von RenderX hat keine Probleme dabei, wenn dieses Attribut fehlt.

Im Gegensatz zum Antenna House XSL Formatter, gibt es auch hier ein paar Probleme bei den einzelnen Linienausprägungen. Werteangaben, die durch das Attribut `border-<Bereich>-style` gesetzt und umgesetzt werden, sind *solid*, *none*, *hidden*, *dotted*, *dashed*, *groove* und *ridge*.

Inzeilige Formatierungen

Das Element `fo:inline` kommt meistens zum Einsatz, wenn spezielle Zeichen, Wörter oder Wortfolgen besonders hervorgehoben werden sollen. Beispielsweise wenn es um Schriftstärken oder Formatierungen einzelner Wörter geht. Jedes Attribut, außer `margin`, das für das Element `fo:block` gilt, ist ebenso für dieses Element gültig. Wenn man beispielsweise ein Wort in der Schriftart *Symbol* haben wollte, so kann dies in der folgenden Struktur gestaltet werden.

```
<fo:inline font-family="Symbol">
    hier der Font in Symbol.
</fo:inline>
```

Fig. 4-8 : Inzeilige Formatierung mit XSL-FO

Über das Attribut `baseline-shift` können einzelne Wörter hoch- oder tiefgestellt werden. Positive Werte entsprechen einer Hochstellung des jeweiligen Wortes, negative Werte einer Tiefstellung.

Apache FOP

Apache FOP unterstützt die Schriftart *Symbol*. Jegliche Auszeichnungen dieser Art werden im Ergebnisdokument richtig umgesetzt.

Jedoch gibt es ein Problem bei der Hoch- bzw. Tiefstellung einzelner Wörter. Angaben betreffend des Attributs `baseline-shift` werden noch nicht unterstützt und bleiben nach der Formatierung ohne Bedeutung. Ein Lösungsansatz, der die jeweiligen Wörter hoch- oder tiefstellt, zeigt folgendes Beispiel.

```
<fo:inline baseline-shift="super">1</fo:inline>
```

Fig. 4-9 : Hoch und Tiefstellung mit baseline-shift

Um Wörter hoch oder tief zu stellen, muss nicht zwingend ein positiver oder negativer Wert hierfür angegeben werden. Zum Beispiel lässt sich ein Wort durch den Wert *super* nach oben versetzen. Um ein Wort tiefzustellen muss der Wert *sub* angegeben werden. Dieser Ansatz wird von FOP unterstützt.

Antenna House XSL Formatter

Der Formatter hat Probleme bei der Fontumstellung. Wenn es darum geht, die Wortfolge in *Symbol* anzuzeigen, sollte sich der Inhalt eines Elements schon in der Schriftart *Symbol* befinden. Bei der Umsetzung dieser Tatsache müsste dies im XSL-Stylesheet über `<xsl:value-of select="."/>` gelöst werden.

Bei der Hoch- bzw. Tiefstellung von einzelnen Wörtern hat der Formatter keine Probleme. Dies kann entweder über positive oder negative Werte im Attribut `baseline-shift` erreicht werden, oder aber auch durch die Werte *sub* und *super*. Dies wird richtig im Ergebnisdokument formatiert und ausgegeben.

RenderX XEP

Der FO-Prozessor XEP unterstützt alle genannten Attribute und setzt diese ordnungsgemäß im Ergebnisdokument um.

Grafiken

Mit XSL-FO ist es möglich, Grafiken, die lokal im System vorhanden sind, in das FO-Dokument einzubinden. Dies wird durch das Element `fo:external-graphic` erreicht. Über das Attribut `src` kann der Pfad zur Grafik referenziert werden. Um die Höhe und Breite der Grafik anzupassen, werden die Attribute `content-height` und `content-width` verwendet. Die Werteangaben sind nicht zwingend. In diesem Fall würde der Wert `auto` indirekt gesetzt und die Grafik in ihren eigentlichen Maßen angezeigt werden. Es sind durchaus auch Werte in Millimeter oder Zentimeter, sowie Prozentangaben möglich, um die Proportionen der Grafik zu bestimmen. [PIKR04]

Apache FOP

Werden im FO-Dokument die Angaben `content-height` und `content-width` beim Einbinden einer Grafik mit angegeben, so werden diese Eigenschaften beim Übertragen in das Ergebnisdokument nicht unterstützt. Beim Angeben dieser Werte gibt es keine sichtbaren Veränderungen im Ergebnisdokument.

Es gibt andere Möglichkeiten, um die Höhe und Breite einer Grafik für das Ergebnisdokument anzupassen. Eine etwas umständlichere Variante ist das Kapseln des Elements `fo:block-container`. Hier können Breiten- und Höhenangaben über die Attribute `width` und `height` gesteuert werden. Jedoch werden hier noch weitere Angaben benötigt, um die Position der Grafik eindeutig zu bestimmen. Dies betrifft die Werte `left` (Abstand von links), `top` (Abstand von oben), und `position` (Angaben für die Positionierung, bspw. den Wert `absolute`). Die untenstehende Abbildung verdeutlicht den Aufwand nochmals.

```
<fo:block-container width="..." height="..."
                    top="..." left="..." position="...">
    <fo:block>
        <fo:external-graphic src="..."/>
    </fo:block>
</fo:block-container>
```

Fig. 4-10 : Block-Container für Grafiken in XSL-FO

Eine andere Variante ist die beiden Attribute `width` und `height` direkt am Element `fo:external-graphic` mit anzugeben. Dies ist wesentlich einfacher als die erste Möglichkeit, da die Angaben für die Positionierung der Grafik direkt über das Attribut `text-align` des Elements `fo:block` angegeben werden können. Hier der Ansatz.

```
<fo:block align="...">
    <fo:external-graphic src="..."
                width="..." height="..."/>
</fo:block>
```

Fig. 4-11 : Grafiken mit width und height in XSL-FO

Antenna House XSL Formatter

Der Formatter von Antenna House unterstützt die Attributsangaben für die Höhe und Breite einer Grafik und setzt diese im Ergebnisdokument richtig um. Man hat die Möglichkeit zur Anpassung der Höhe und Breite die Attribute `content-height`, `content-width` oder nur `width` und `height` anzugeben. Es wird jede Variante unterstützt.

RenderX XEP

Der FO-Prozessor XEP verhält sich bei der Umsetzung der Varianten gleich wie der Antenna House XSL Formatter.

Blöcke

Die meisten Elemente werden durch das Element `fo:block` umschlossen. Hier können Angaben bezüglich der Schriftart, Schriftgröße, Schriftgrad, Ausrichtungen, Hintergründe, seitliche Abstände usw. getroffen werden, die sich auf den gesamten Inhalt des Blocks auswirken. Beispielsweise kann man den Abstand von links durch das Attribut `margin-left` beeinflussen. In der Diplomarbeit sollen nur die Interpretationen von Zentimeter- , Millimeter- und Prozentangaben untersucht werden.

Um den Abstand einzelner Ziffern oder Wörter zu erhöhen bzw. zu verringern, definiert man dies in den Attributen `letter-spacing` und `word-spacing`. Man kann auch Wörter und Sätze in Groß- bzw. Kleinbuchstaben transformieren, in dem man das Attribut `text-transform` auf die Werte `uppercase` bzw. `lowercase` setzt. Um einzelne Zeilen anzuordnen, können die Attribute `text-align` und `text-align-last` (mit Bezug auf die letzte Zeile) verwendet werden. Um speziell einen Einfluss auf die Silbentrennung eines Absatzes zu haben, gibt es die Attribute `hyphenate` und `hyphenation-character`, um die Silbentrennung zu erlauben bzw. um das jeweilige Trennzeichen zu setzen.

Apache FOP

Apache FOP hat Probleme bei der Darstellung von Blöcken. So kann das Attribut `margin-left` nur Zentimeter- oder Millimeterangaben interpretieren. Prozentangaben werden ignoriert.

Probleme bereiten weiterhin das Vergrößern von Wortabständen durch `word-spacing` sowie die Formatierung über die Attribute `text-transform` und `text-align-last`. In der Ausspielung des Ergebnisdokuments werden Standardeinstellungen verwendet.

Die Silbentrennung wird mit FOP korrekt umgesetzt. Als Trennzeichen wird jedoch standardmäßig ein Bindestrich verwendet. Der Anwender kann somit kein individuelles Trennzeichen vergeben.

Antenna House XSL Formatter

Der Formatter von Antenna House hat keine Probleme bei der Umsetzung seitlicher Abstände. Ebenso kann er mit Zeichen- und Wortabständen, Groß- und Kleinschreibungen sowie Textausrichtungen umgehen und setzt diese ordnungsgemäß im Ergebnisdokument um. Auch die Silbentrennung wird vom Formatter richtig interpretiert und umgesetzt.

RenderX XEP

Ebenso wie der Formatter hat auch XEP keine von den genannten Problemen des Apache FOP.

Horizontale Linien

Das Element `fo:leader` wird eingesetzt, um horizontale Linien darzustellen. Wie bei den Rändern einer Tabelle können diese Linien gepunktet, gestrichelt oder durchgehend dargestellt werden. Ebenso ist es möglich, über selbst definierte Zeichen eine Linie zu definieren. Über das Attribut `rule-style` kann der Stil einer Linie bestimmt werden, bspw. *dotted*. Dazu muss das Attribut `leader-pattern` (die Füllzeichen einer Linie) auf den Wert `rule` gesetzt werden. [PIKR04] Im Folgenden ein kleines Beispiel.

```
<fo:leader leader-length="..."
           rule-thickness="..."
           rule-style="dotted"
           leader-pattern="rule"/>
```

Fig. 4-12 : Linien in XSL-FO mit fo:leader

Apache FOP

Apache FOP unterstützt nicht jedes Linienmuster. Die einzigen Muster, die richtig dargestellt werden können sind *dotted*, *dashed*, *solid* und *double*. Weiterhin kann es sein, dass die Abstände, bspw. bei doppelten Linienführungen im Gegensatz zu den anderen Prozessoren viel geringer sind. Dies hängt aber von der jeweiligen Formatierung des Interpreters ab.

Antenna House XSL Formatter

Der FO-Prozessor von Antenna House kann jegliche Linienmuster unterstützen. Das sind *dotted*, *dashed*, *solid*, *double*, *groove* und *ridge*.

RenderX XEP

Ebenso wie der Formatter kann auch XEP jegliche Linienmuster unterstützen und im Ergebnisdokument darstellen.

Das Float-Konzept

So genannte Float-Objekte werden benötigt, wenn bspw. Grafiken von Texten umschlossen werden sollten. Das Objekt, dass umflossen werden soll, wird in das Element `fo:float` aufgenommen. Meistens sind diese Objekte in einem Block-Objekt definiert. Das Attribut `float` gibt an, wo das zu umfließende Objekt platziert werden soll. Wenn das Objekt an den rechten Rand versetzt werden soll, so nimmt dieses Attribut den Wert *right* an. Zum verdeutlichen der Probleme bei der Umsetzung durch die FO-Prozessoren wird folgendes Beispiel angegeben.

```
<fo:block>
    TEXT, TEXT ...
    <fo:float float="left">
        <fo:block>
            <fo:external-graphic src="..."/>
        </fo:block>
    </fo:float>
    TEXT, TEXT ...
</fo:block>
```

Fig. 4-13 : Float-Objekte in XSL-FO nach [PIKR04]

Hier wird eine Grafik als Float-Objekt definiert. Die Ausrichtung des Objekts ist am linken Rand deklariert worden. Um das Objekt selbst fließen Texte.

Apache FOP

Die obige Umsetzung wird über den FO-Prozessor FOP nicht umgesetzt. Das Problem ist, dass Float-Objekte noch nicht unterstützt werden. Die obige angegebene Grafik wird nicht im Ergebnisdokument dargestellt.

Antenna House XSL Formatter

Float-Objekte werden durch den Formatter von Antenna House unterstützt. Die Grafik befindet sich am linken Rand des Ergebnisdokuments. Der Text umfließt dieses und endet mit dem letzten Wort.

RenderX XEP

Ebenso wie Antenna House unterstützt dieser FO-Prozessor das Float-Konzept.

4.4 Grafische Unterschiede zwischen den FO-Prozessoren

Im Vergleich fallen neben der weiter vorn beschriebenen Probleme maßgebliche Unterschiede zwischen den Prozessoren auf. Unterschiede betreffen die Darstellungen der einzelnen Objekte auf einer Seite. Auch hier gibt es Elemente bzw. Attribute, auf die näher eingegangen wird, um den Unterschied zwischen diesen deutlicher hervorzuheben.

4.4.1 Seiteninhalte

Das Element `fo:page-sequence` enthält Kinder, die für die Darstellung des Inhalts von Bedeutung sind. Die folgenden Elemente haben einige Abweichungen in ihrer Form und Darstellung.

Fußnoten

Mit dem Element `fo:footnote` können Fußnoten erzeugt werden. Der eigentliche Kontext der Fußnote wird mit dem Element `fo:footnote-body` gesetzt. Ein wesentlicher Nachteil bei den Fußnoten liegt darin, dass man nicht direkt angeben kann, wo der Inhalt einer Fußnote platziert werden soll. Dieser wird automatisch am Ende einer Seite, vor der Fußzeile, platziert.

Apache FOP

Der Inhalt der Fußnote beginnt mit Apache FOP direkt hinter der jeweiligen Nummerierung, sprich auf der selben Grundlinie. Dies ist jedoch nicht der Standard, der bei den meisten Fußnoten zu finden ist. Normalerweise ist die Nummerierung hochgestellt. Die Darstellung einer Fußnote mit Apache FOP zeigt die nachfolgende Abbildung.

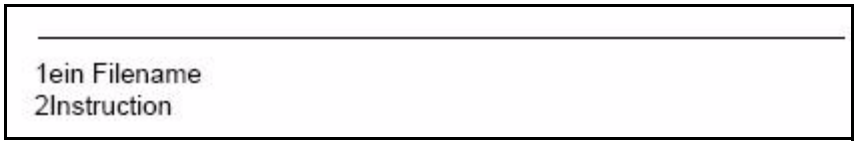


Fig. 4-14 : Darstellung einer Fußnote mit Apache FOP

Antenna House XSL Formatter

Der Antenna House XSL Formatter formatiert den Inhalt etwas unter der Nummerierung der Fußnote. So kommt der Eindruck auf, dass die Nummerierung etwas hochgestellt platziert wurde, wie es auch im eigentlichen Inhalt der Seite hinter einem Wort gekennzeichnet wurde. Die folgende Abbildung verdeutlicht diesen Unterschied zu Apache FOP.

Fig. 4-15 : Darstellung einer Fußnote mit Antenna House XSL Formatter

RenderX XEP

Der FO-Prozessor setzt die Fußnoten genauso wie der Formatter von Antenna House um. Dies wird bei der unteren Abbildung nochmals vergegenwärtigt.

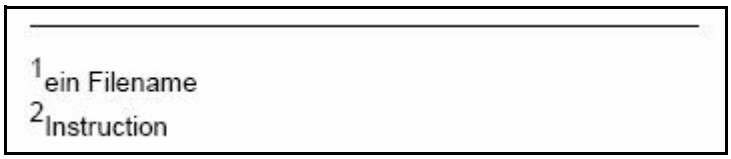


Fig. 4-16 : Darstellung einer Fußnote mit RenderX XEP

Tabellenstrukturen

Hierbei gibt es große Unterschiede bei den Elementen `fo:table-column`, `fo:table-row`, und `fo:table-cell`, welche jeweils für die Definition einer Spalte, einer Reihe und einer Zelle bestimmt werden. Folgendes Beispiel soll einmal gegeben sein.

```
<fo:table>
    <fo:table-column column-width="1cm"/>
    <fo:table-column column-width="100%"/>
    <fo:table-column column-width="1cm"/>
    <fo:table-body>
        <fo:table-row>
            <fo:table-cell>
                <fo:block text-align="left">
                    <xsl:number .../>
                </fo:block>
            </fo:table-cell>
            <fo:table-cell>
                <fo:block text-align="justify">
                    <fo:leader ... />
                </fo:block>
            </fo:table-cell>
            <fo:table-cell>
                <fo:block text-align="end">
                    <xsl:page-number-citation .../>
                </fo:block>
            </fo:table-cell>
        </fo:table-row>
    </fo:table-body>
</fo:table>
```

Fig. 4-17 : Tabellenstruktur in XSL-FO

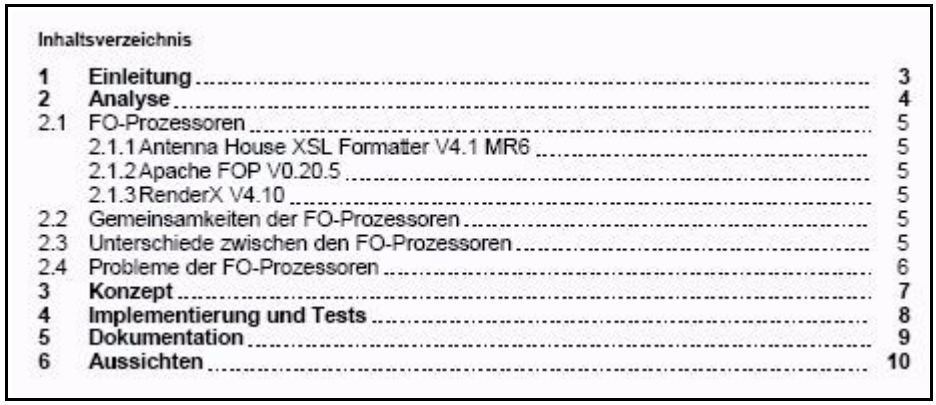
In diesem Beispiel soll ein Inhaltsverzeichnis angelegt werden, welches immer aus drei Spalten und einer Zeile besteht. Dabei definieren sich die Spalten wie folgt: Die erste Spalte hat eine Breite von einem Zentimeter. Der Inhalt dieser Spalte gibt die jeweilige Kapitelnummer wieder. Diese wurde hier durch `xsl:number` bestimmt. Die zweite Spalte wurde mit hundert Prozent angegeben. In diesem Fall würde diese Spalte, die restliche Breite der Tabelle annehmen. Der Inhalt dieser Spalte gibt den eigentlichen Kontext aus. In dem Beispiel, welches nicht ganz dargestellt wurde, befindet man sich an dem jeweiligen Elementknoten. Somit kann über `xsl:value-of` der Wert dessen ausgegeben werden. Nach dem die Überschrift des jeweiligen Kapitels ausgegeben wurde, wird der Rest durch eine punktierte Linie aufgefüllt. Dies wurde mit `fo:leader` darge-

stellt. Die letzte Spalte wurde wieder mit einem Zentimeter angegeben. Diese bezieht nun die jeweilige Seitenzahl des angefangenen Kapitels über `fo:page-number-citation`.

Das Beispiel soll zeigen, wie eine Tabelle aufgebaut sein kann, um diese mit den drei FO-Prozessoren Antenna House XSL Formatter, Apache FOP und RenderX XEP in das Ausgabeformat PDF zu überführen. Somit soll der Unterschied nochmals verdeutlicht werden.

Apache FOP

Die Tabelle wird wie erwünscht formatiert. Die untenstehende Abbildung zeigt dies nochmals deutlicher.



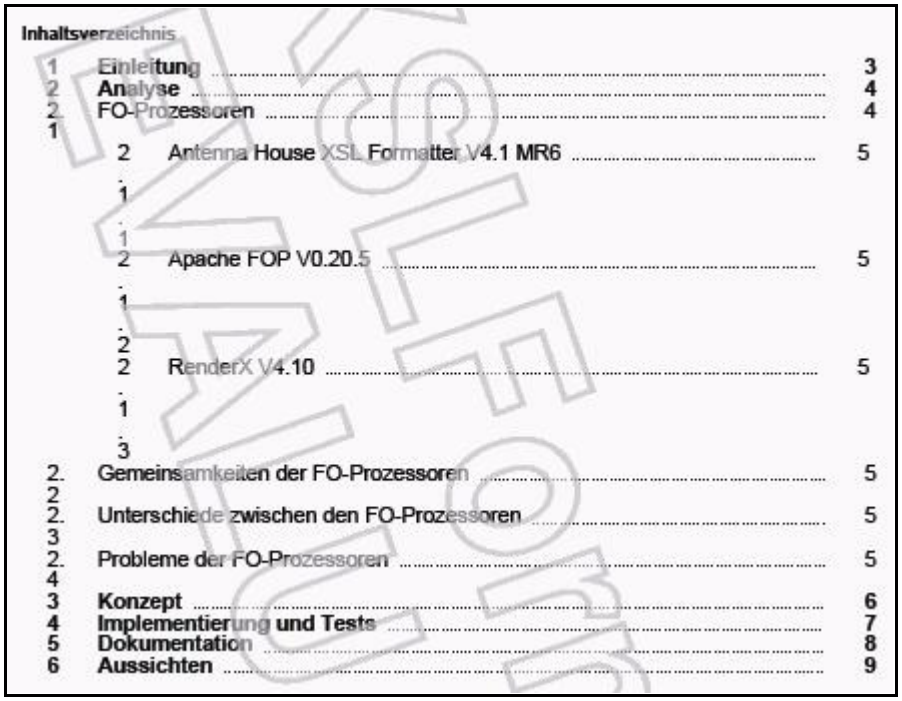
Inhaltsverzeichnis

Fig. 4-18 : Darstellung einer Tabelle mit Apache FOP

Hier sieht man deutlich, wie die einzelnen Spalten ausgegeben werden. Die erste und letzte Spalte beziehen sich jeweils auf einen Zentimeter. Die mittlere Spalte umfasst hundert Prozent, sprich den restlichen Teil der Tabelle, um die Tabellenbreite auszufüllen. Die Einrückung der jeweiligen Tabelle, vgl. Kapitelnummer 2.1.1, ist individuell gestaltet worden.

Antenna House XSL Formatter

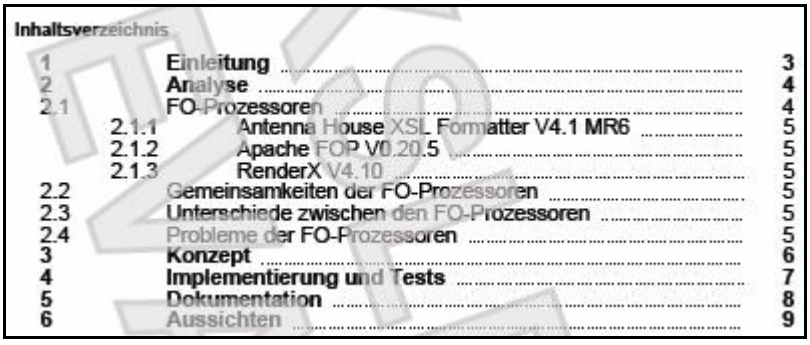
Der Antenna House XSL Formatter formatiert die Tabelle etwas anders und verzieht die Spalten in ihrer seitlichen Ausdehnung. Man kann dies in der folgenden Abbildung erkennen.


Inhaltsverzeichnis

1 Einleitung 3
2 Analyse 4
2.1 FO-Prozessoren 4
2.1.1 Antenna House XSL Formatter V4.1 MR6 5
2.1.2 Apache FOP V0.20.5 5
2.1.3 RenderX V4.10 5
2.2 Gemeinsamkeiten der FO-Prozessoren 5
2.3 Unterschiede zwischen den FO-Prozessoren 5
2.4 Probleme der FO-Prozessoren 5
3 Konzept 6
4 Implementierung und Tests 7
5 Dokumentation 8
6 Aussichten 9

Fig. 4-19 : Darstellung einer Tabelle mit Antenna House XSL Formatter

Wie man hier erkennen kann, verzieht sich die erste Spalte, da beispielsweise das Kapitel mit der Nummer 2.1 wohl breiter, als die fest angegebene Spaltenbreite, angegeben mit einem Zentimeter, ist. Ebenso ist der restliche Teil einer Zeile etwas verzogen. Die Seitenzahlen sind nicht auf der selben Höhe wie bei Apache FOP. Dies muss individuell im XSL-Stylesheet für den FO-Prozessor angepasst werden. Die erste Spalte sollte eine Breite von 2,5 cm, die zweite Spalte 80 Prozent und die letzte Spalte 1 cm haben. So kommt man der Ausgabe von Apache FOP schon etwas näher. Jedoch gelingt es nicht ganz, dieselbe Darstellung zu erreichen. Man kann die Werte der einzelnen Spaltenbreiten so ändern, dass diese einheitlich in einer Zeile stehen würden. Dennoch gelingt es nicht, die Darstellung der Tabelle wie in der Abbildung Fig. 4-18 umzusetzen. Das beste was hierbei durch die Anpassung der Spaltenbreite erzeugt werden kann, sieht man in der nachfolgenden Abbildung.


Inhaltsverzeichnis

1 Einleitung 3
2 Analyse 4
2.1 FO-Prozessoren 4
2.1.1 Antenna House XSL Formatter V4.1 MR6 5
2.1.2 Apache FOP V0.20.5 5
2.1.3 RenderX V4.10 5
2.2 Gemeinsamkeiten der FO-Prozessoren 5
2.3 Unterschiede zwischen den FO-Prozessoren 5
2.4 Probleme der FO-Prozessoren 5
3 Konzept 6
4 Implementierung und Tests 7
5 Dokumentation 8
6 Aussichten 9

Fig. 4-20 : Bessere Darstellung einer Tabelle durch den Formatter

Um die Tabelle so aussehen zu lassen, bedarf es einer kleinen Änderung bei der Definition der Spalten.

```
<fo:table>
    <fo:table-column column-width="2.5cm"/>
    <fo:table-column column-width="80%"/>
    <fo:table-column column-width="1cm"/>
    <fo:table-body>
        ...
    </fo:table-body>
</fo:table>
```

Fig. 4-21 : Spaltenanpassung bei Antenna House XSL Formatter

RenderX XEP

XEP interpretiert die Tabelle ebenso wie Apache's FOP und stellt diese in der folgenden Form dar.

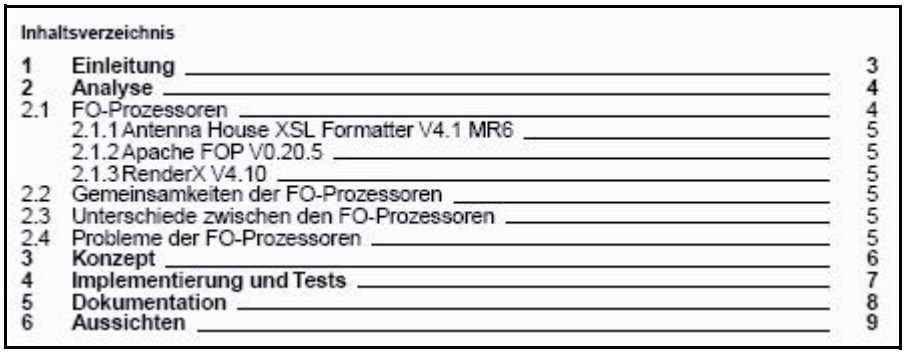


Fig. 4-22 : Darstellung einer Tabelle mit RenderX XEP

Hier sieht man deutlich, wie die Spalten mit ihren Inhalten umgesetzt wurden.

Listen und Aufzählungen

Mit XSL-FO lassen sich geordnete, ungeordnete und Beschreibungslisten durch das Element `fo:list-block` darstellen. Geordnete Listen beginnen meistens mit einer fortlaufenden Nummerierung, im Gegensatz zu ungeordneten Listen, welche durch ein Symbol am Anfang der Aufzählung gekennzeichnet werden. Mit Beschreibungslisten sind Listen gemeint, die durch eine Beschreibung des Inhalts gekennzeichnet sind. Die Liste selbst besteht aus mehreren Listenpunkten, welche über das Element `fo:list-item` handhabbar sind. Das sogenannte Etikett einer Liste wird durch `fo:list-item-label` dargestellt. Der eigentliche Inhalt der Liste wird in das Element `fo:list-item-body` aufgenommen.

Apache FOP

Hier gibt es bei der Überführung des FO-Dokuments in das jeweilige Ausgabeformat kleine Abweichungen gegenüber den anderen beiden Prozessoren. Die Inhalte der Liste sind im Gegensatz zur Nummerierung oder zu den Aufzählungszeichen etwas nach unten hin verschoben worden, siehe folgenden Abbildung.

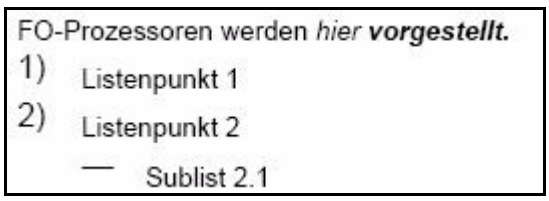


Fig. 4-23 : Listendarstellung mit Apache FOP

Antenna House XSL Formatter

Bei der Überführung des Formatters sind die Listeninhalte hinter den Aufzählungszeichen nach oben hin verschoben worden, jedoch nur um Millimeter. Sie sind schon fast auf der gleichen Zeilenhöhe. Diese kleinen Unterschiede können jedoch vernachlässigt werden.

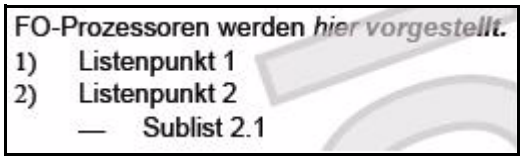


Fig. 4-24 : Listendarstellung mit Antenna House XSL Formatter

RenderX XEP

Wie auch beim Formatter von Antenna House werden hier die Inhalte einer Liste nach oben hin verschoben. Hier sieht man schon deutlicher, dass die Listenpunkte nicht auf der Grundlinie der Nummerierung liegen.

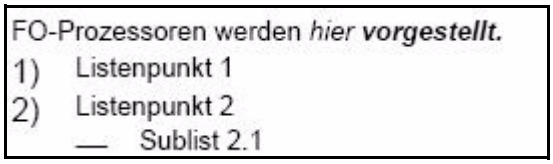


Fig. 4-25 : Listendarstellung mit RenderX XEP

Blöcke

Die Schriftart kann in einem Block über das Attribut `font-family` gesetzt werden. Meistens wird die Schriftgröße, `font-size`, noch mit angegeben. Hier ein kleines Beispiel dafür, wie man die Schriftart *Arial* und die Schriftgröße *8pt* für einen Block definiert.

```
<fo:block font-family="Arial" font-size="8pt">
    ein einfaches Beispiel
</fo:block>
```

Fig. 4-26 : Schriftgestaltung in fo:block

Apache FOP

Um die jeweilige im FO-Dokument definierte Schriftart in das Ergebnisdokument zu übertragen, bedarf es einer Konfigurations-Datei, welche auf die jeweilige Schriftart verweist. Sollten diese Dateien nicht bekannt sein, gibt eine Default-Schriftart an. Im Folgenden ein Beispiel, wie eine Konfigurations-Datei aussehen kann.

```
<configuration>
    <fonts>
        <font metrics-file="arial.xml" embed-file="arial.ttf" >
            <font-triplet name="Arial"  />
            <font-triplet name="ArialMT" />
        </font>
    </fonts>
</configuration>
```

Fig. 4-27 : Konfigurationsdatei für Font Arial

Antenna House XSL Formatter

Hier bedarf es keiner Konfigurationsdatei, um den jeweiligen Font dem FO-Prozessor bekannt zu machen. Die angegebene Schriftart wird richtig im Ergebnisdokument ausgespielt.

RenderX XEP

Auch RenderX XEP benötigt keine Konfigurationsdatei zum Auffinden des gesetzten Fonts im FO-Dokument. Jedoch gibt es kleine Unterschiede in der Bezeichnung der Schriftarten. Beispielsweise kennt Antenna House XSL Formatter die Bezeichnung *Courier New*, während XEP hier einen Fehler über die Konsole ausgibt, dass der gesetzte Font nicht gefunden werden kann. XEP erkennt den Font nur unter der Bezeichnung *Courier*.

4.4.2 Der Seitenaufbau

Im folgenden werden Unterschiede in den einzelnen Bereichen eines Druckbereichs auf ihre unterschiedlichen Darstellungen hervorgehoben. Diese betreffen vor allem die Bereiche Kopfbereich (`fo:region-before`), Fußbereich (`fo:region-end`), linker (`fo:region-start`) und rechter (`fo:region-end`) Seitenbereich.

Der Druckbereich

Es ist möglich, Ränder um die einzelnen Bereiche zu zeichnen. Diese Ränder beziehen sich auf die oberen, unteren, linken und rechten Grenzen. Sie können schon bei der Definition einer Master-Page angelegt werden. Im unteren Beispiel wurde eine Linie am unteren Rand des linken Seitenbereichs definiert. Bei dieser Untersuchung wurde speziell auf das Attribut `border-bottom-width` eingegangen, um den Unterschied in der Darstellung einer Linie in Bezug auf ihre Linienstärke herauszufiltern. Das Beispiel wurde in Bezug auf alle drei FO-Prozessoren getestet.

```
<fo:region-start margin="10mm 2mm 10mm 2mm"
                 border-bottom-style="solid"
                 border-bottom-color="black"
                 border-bottom-width="2pt"/>
```

Fig. 4-28 : Linie am unteren Rand einer Tabelle mit fo:region-start

Dabei ist die Linienart eine durchgezogene schwarze Linie mit der Stärke von zwei Punkten. Hierbei ergaben sich nun folgende Unterschiede beim Ausführen der einzelnen FO-Prozessoren.

Apache FOP

Wie bereits in den Problemen zuvor berichtet, unterstützt der Apache FOP diese Attribute nicht und kann somit keine Ränder in diesen Bereichen zeichnen. Jedoch können Ränder über das Anlegen einer Tabelle im Element `fo:static-content` definiert werden. Hierzu gibt es ebenso Attribute wie `border-bottom-style` (vgl. hierzu Kapitel "Der Druckbereich").

Antenna House XSL Formatter

Der Antenna House XSL Formatter hat kleinere Abweichungen bei der Darstellung solcher Linien. Wenn man zuvor diese Einstellungen über den Prozessor XEP laufen lässt, so ist die Linie, trotz gleichen Werten, beim Formatter wesentlich dicker als bei RenderX XEP. Um nun die gleiche Stärke wie von XEP im Ergebnisdokument zu erlangen, muss diese Stärke um 0,5 Einheiten nach unten verändert werden. Somit kommt man dem Ergebnis wesentlich näher. Wie der Formatter dieses Beispiel darstellt, zeigt die folgende Abbildung.

Fig. 4-29 : Linienstärke bei Antenna House XSL Formatter

RenderX XEP

Wie auch schon beim Formatter erwähnt, ist die Stärke der Linie in den Bereichen hier um 0,5 Einheiten schwächer. Es unterliegt dem Anwender, wie er dies handhaben möchte. Die folgende Abbildung zeigt den Unterschied zum Formatter von Antenna House.

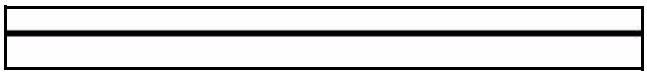


Fig. 4-30 : Linienstärke bei RenderX XEP

4.5 Gemeinsamkeiten der FO-Prozessoren

In diesem Abschnitt der Diplomarbeit werden die Gemeinsamkeiten der zuvor vorgestellen FO-Prozessoren herauskristallisiert. Da XSL-FO unzählige Elemente und Attribute anbietet, werden hier nur die wesentlichen Elemente zusammengefasst, die meistens in einem FO-Dokument zu finden sind und die durch jeden der drei FO-Prozessoren gleich umgesetzt werden. Dabei sollten sich Gemeinsamkeiten in der Definition einer Master-Page, sowie in dem Zuweisen von Inhalten einer Seitenfolge aufweisen. Betrachtet wird im Folgenden der Seitenaufbau.

4.5.1 Der Seitenaufbau

Wie schon in dem Kapitel 4.3.1 verdeutlicht wurde, ist das wesentliche Element, um eine Seite zu definieren, das `fo:simple-page-master`. Hier lassen sich die Abstände nach oben, unten, links und rechts über das Attribut `margin` steuern. Im folgenden ein Beispiel.

```
<fo:simple-page-master margin="10mm 7mm 18mm 7mm" >
    <fo:region-body .../>
    ...
</fo:simple-page-master>
```

Fig. 4-31 : fo:simple-page-master mit Attribut margin

Dies bedeutet, dass ein Rand mit dem Abstand nach oben 10 mm, nach rechts 7 mm, nach unten 18 mm und nach links 7mm für diese Seite definiert wird. Dies ist jedoch eine kürzere Schreibweise. Man kann dies auch durch die Attribute `margin-top`, `margin-bottom`, `margin-left` und `margin-right` erreichen. Dies würde dann folgendermaßen aussehen.

```
<fo:simple-page-master margin-top="10mm"
                       margin-right="7mm"
                       margin-bottom="18mm"
                       margin-left="7mm">
    <fo:region-body .../>
    ...
</fo:simple-page-master>
```

Fig. 4-32 : fo:simple-page-master 2. Möglichkeit mit margin

Einen kurzen Überblick darüber, welche Auswirkungen die Attribute `margin` auf eine Seite haben, soll folgende Grafik zeigen.

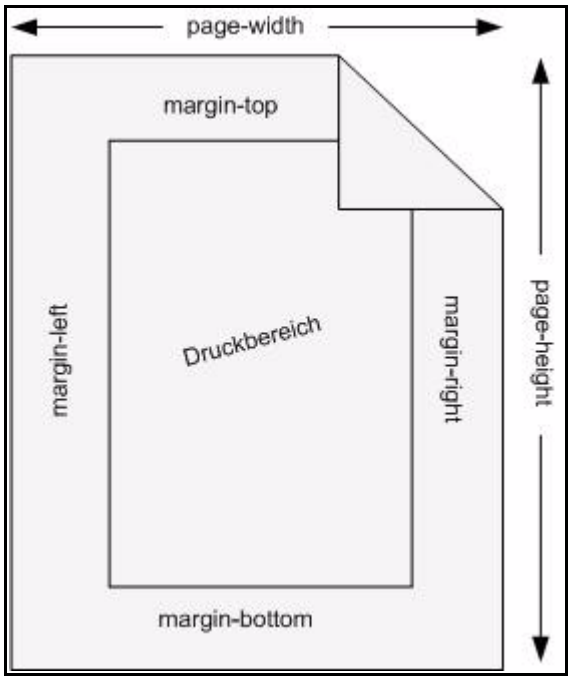


Fig. 4-33 : Seitenaufbau des fo:simple-page-master nach [PIKR04]

Damit das Layout einer Seite komplett ist, werden die Attribute `page-height` und `page-width` verwendet. Somit kann die Höhe und Breite einer Seite angegeben werden. Wenn diese Angaben nun in das FO-Dokument mit einfließen, so werden diese bei allen drei Prozessoren gleichermaßen umgesetzt. Diese Gemeinsamkeit macht es einfacher, unterschiedliche FO-Prozessoren bei der Überführung eines Ergebnisdokuments zu verwenden. Würden hier Probleme bei den Darstellungen auftreten, so würde dies bedeuten, dass es unterschiedliche Konfigurationen geben müsste, oder jedesmal die Werte der Anwendung verändert werden müssten, um die jeweilige korrekte Darstellung durch alle drei FO-Prozessoren zu erreichen. Diese Angaben, ohne Höhe und Breite der Seite, gelten auch für das Element `fo:region-body`, welches die Ränder für den Hauptbereich festlegt.

Der Druckbereich

Der Inhalt einer Seite kann nochmals unterteilt werden in Kopfzeile (`region-before`), Fußzeile (`region-after`), linkem Seitenbereich (`region-start`) und rechtem Seitenbereich (`region-end`). Diese Unterteilung soll in der nachfolgenden Abbildung genauer veranschaulicht werden, um eine bessere Vorstellung der jeweiligen Eingrenzungen zu bekommen.

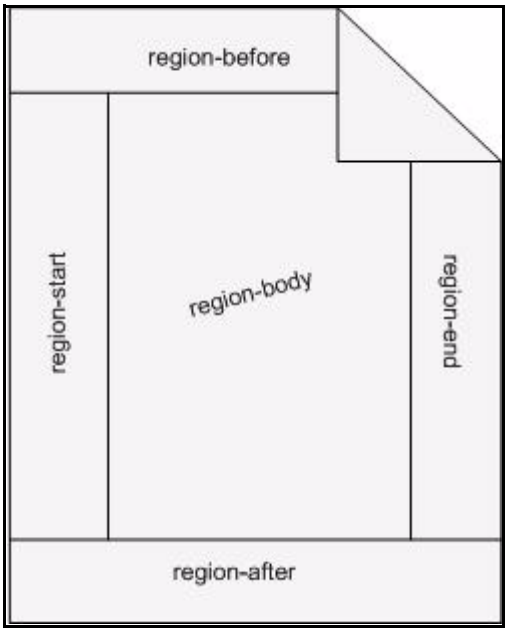


Fig. 4-34 : Bereiche des fo:simple-page-master nach [PIKR04]

Um den Rand dieser Bereiche, außer dem Hauptbereich (gekennzeichnet durch das Element `fo:region-body`), zu verändern, bedarf es des Attributs `extent`. Ein Beispiel für die Kopfzeile könnte wie folgt aussehen.

```
<fo:simple-page-master margin="...">
    <fo:region-before extent="2mm" .../>
    ...
</fo:simple-page-master>
```

Fig. 4-35 : Beispiel einer Kopfzeile mit Attribut extent

Dies würde nun bedeuten, dass der Abstand der Kopfzeile zum eigentlichen Hauptbereich von oben herab kommend, um 2 mm herabgesenkt wäre. Auch hier sind die Auswirkungen bei allen drei Prozessoren gleich, was die anderen Bereiche betrifft. Durch die Auswahl eines beliebigen Prozessors würde sich die Darstellung jedesmal gleich verhalten.

4.5.2 Blöcke

Eines der wohl wichtigsten Elemente einer Seite ist das Block-Element `fo:block`. Ein Block ist ein rechteckiger Bereich, in dem Grafiken, Tabellen, Listen, usw. eingesetzt werden können. Es ist auch möglich, dass Blöcke ineinander verschachtelt werden. Um den Inhalt eines Blocks zu gestalten, können Attribute verwendet werden. Wie auch in Kapitel 4.5.1 schon beschrieben, hat ein Block die Möglichkeit sich einzugrenzen durch die Attribute `margin-top`, `margin-bottom`, `margin-left` und `margin-right`. Diese Auswirkungen prägen sich bei allen drei FO-Prozessoren genau gleich aus.

Ebenso kann einem Block eine gewisse Hintergrundfarbe durch die Eigenschaft `background-color` hinzugefügt werden. Auch hier können gewisse Werte eingetragen werden, wobei sich die Farbe im Ergebnisdokument bei jedem FO-Prozessor gleich verhält. Anstatt der Hintergrundfarbe können auch Wörter bzw. Inhalte eines Blocks in gewissen Farben verwaltet werden. Dies kann mit dem Attribut `color` erreicht werden. Auch hier wird dies von allen drei Prozessoren unterstützt.

Es ist weiterhin möglich, Innenabstände eines Blocks zu definieren. Dies geschieht durch die sogenannten Padding-Attribute. Diese sind u.a. `padding-left`, `padding-right`, `padding-top` und `padding-bottom`. Das hat zur Folge, dass sich die jeweiligen Abstände nach links, rechts, oben und unten erstrecken und so einen Block verengen bzw. weiten können. Diese Änderungen sind bei den FO-Prozessoren ohne Unterschiede festzustellen.

Eine weitere Gemeinsamkeit ist die Ausrichtung eines Blocke nach links, rechts und zentriert, welche sich über das Attribut `text-align` regeln lassen. All die genannten Attribute eines Blockes sind die gängigsten, die in den meisten Dokumenten zu finden sind.

4.5.3 Grafiken

Bei den Grafiken gibt es in manchen Hinsichten ein paar Probleme, jedoch lassen sich auch Grafiken definieren, die durch jeden FO-Prozessor gleich umgesetzt werden. Die Definition einer Grafik wird durch das Element `fo:external-graphic` bestimmt. Der Pfad zur Grafik wird im Attribut `src` festgelegt. Um Grafiken in ihrer Höhe und Breite zu verändern kann dies über zwei Wege erfolgen. Wenn die Attribute `content-height` und `content-width` verwendet werden, so kann dies vom Apache FOP nicht unterstützt werden, da er dieses nicht interpretieren kann. Jedoch können alle drei FO-Prozessoren mit den Attributen `width` und `height` umgehen. Diese bestimmen eine Grafik ebenso in ihrer Höhe und Breite.

4.5.4 Querverweise

Was in einem PDF-Dokument ebenso häufig vorkommt sind Querverweise bzw. Links. Beispielsweise wenn ein Inhaltsverzeichnis generiert wurde, sollte es möglich sein, per Klick auf das jeweilige Kapitel direkt auf die jeweilige Seite zu springen. Dies ist über das Element `fo:basic-link` möglich. Um einen Link nutzen zu können, z.B. innerhalb eines Dokuments, muss der jeweilige Block, auf den sich der Link bezieht, mit dem Attribut `id` gekennzeichnet werden. [PIKR04] Die nachfolgende Abbildung zeigt wie dies zu verstehen ist.

```
<fo:block ID="10013" ...>
    ....
</fo:block>
```

Fig. 4-36 : Block mit einer ID

Um nun die Verbindung zum Kapitel und dem Link zu erhalten, muss das Attribut `internal-destination` mit dem Wert der ID des Blocks an das Link-Element `fo:basic-link` gesetzt werden. Diese Variante sieht folgendermaßen aus.

```
<fo:block >
    Hier ist ein Link
    <fo:basic-link internal-destination="10013" ...>
        Vgl. Kapitel 1 Einführung.
    </fo:basic-link>
</fo:block>
```

Fig. 4-37 : Querverweis mit fo:basic-link

Hier wird nun auf die ID des obigen Blockes verwiesen. Durch das Attribut `show-destination` kann bei Klick auf den Link zu dem jeweiligen Block gesprungen werden. Dies unterstützen alle drei Prozessoren ohne Probleme.

4.5.5 Horizontale und vertikale Linien in Tabellen

Um horizontale oder vertikale Linien in der Kopfzeile oder in der Fußzeile zu definieren, muss eine Tabelle angelegt werden. Das Element `fo:table-body` besitzt Attribute, um einen Rahmen um die gesamte Tabelle zu zeichnen. Hier können Linien für oben, unten, links und rechts gesetzt werden. Es kann die Linienstärke, Linienfarbe und der Linienstil (Hinweis auf die Probleme bei Liniendarstellungen in Kapitel "Horizontale Linien") angegeben werden. Ein kleines Beispiel zeigt, wie man eine horizontale Linie bspw. in der Kopfzeile durch eine Tabelle definieren kann. Um eine horizontale Linie einzufügen, welche sich unterhalb eines Textes oder einer Grafik befindet, bedarf es des Attributs `border-bottom-style`. Hiermit wird festgelegt, welcher Linientyp dargestellt werden soll. Um nun dieser Linie eine Farbe zu verleihen, wird das Attribut `border-bottom-color` hinzugefügt. Durch `border-bottom-width` kann die Stärke der Linie bestimmt werden. Im unteren Beispiel wäre die Linienstärke 1 pt. Dies ist meistens der Standardwert, der hierfür gesetzt wird.

```
<fo:table>
    <fo:table-column/>
    <fo:table-body border-bottom-style="solid"
                   border-bottom-color="black"
                   border-bottom-width="1pt">
        <fo:table-row>
            <fo:table-cell>...</fo:table-cell>
        </fo:table-row>
    </fo:table-body>
</fo:table>
```

Fig. 4-38 : Horizontale Linien in einer Tabelle

4.5.6 Kapitelüberschriften in Kopfzeilen

Aus Kapitel "Kapitelüberschriften in der Kopfzeile" hat man schon das Element `fo:retrieve-marker` kennengelernt. Um dieses Element anwenden zu können, muss zuvor das Element `fo:marker` eingesetzt werden. Über dieses Element wird in gewisser Hinsicht eine Referenz gesetzt, die im gesamten FO-Dokument bezogen werden kann. Ein kleines Beispiel hierzu.

```
<fo:block>
    <fo:marker marker-class-name="H1_Heading">
        <fo:block>Kapitel Überschrift</fo:block>
    </fo:marker>
</fo:block>
```

Fig. 4-39 : Beispiel mit fo:marker

So können einfach Kapitelüberschriften in Kopfzeilen generiert werden. Hierzu muss am Marker-Objekt das Attribut `marker-class-name` gesetzt werden, dass einen Namen übergeben bekommt. Dieser Name wird dann über `retrieve-class-name` im Element `fo:retrieve-marker` angezogen. Das folgende Beispiel zeigt, wie man eine Kapitelüberschrift beziehen kann, die beim ersten Eintreffen eines neuen Kapitels gesetzt wird. Jedoch wird bei diesem Beispiel die Überschrift nicht für die fortlaufenden Seiten mit überliefert.

```
<fo:retrieve-marker
    retrieve-boundary="page"
    retrieve-position="first-including-carryover"
    retrieve-class-name="H1_Heading"/>
```

Fig. 4-40 : Kapitelüberschriften mit XSL-FO durch retrieve-marker

Bei Einhaltung dieser Struktur sind keine Probleme aufgetreten.

4.6 Gesamtüberblick über die Unterstützungen der FO-Prozessoren

Dieses Kapitel soll in einer Tabelle nochmals die Unterschiede und Problematiken der FO-Prozessoren zusammentragen. Die Gemeinsamkeiten wurden hierbei nicht berücksichtigt, da diese für den weiteren Vortlauf des Lösungswegs nicht so stark von Bedeutung sind.

4.6.1 Unterstützungen des FO-Standards

In der folgenden Tabelle sind nochmals die Unterstützungen des FO-Standards durch die unterschiedlichen FO-Prozessoren in Bezug auf einige XSL-FO-Elemente zusammengetragen worden.

Element	Attribut	Apache FOP	RenderX XEP	Antenna House XSL Formatter
`<fo:region-body />`	`background-image` `background-image-horizontal`	Keine Unterstützung für Grafiken, Höhe und Breite sowie Ausrichtung lassen sich nicht beeinflussen	Unterstützung für Grafiken	Unterstützung für Grafiken
	`border-color`	Keine Unterstützung von Randeinstellungen	Unterstützung von Randeinstellungen	Unterstützung von Randeinstellungen
	`reference-orientation`	Keine Unterstützung für Änderung der Gradzahl, bspw. um 90°	Unterstützung für Änderungen der Gradzahl	Keine Unterstützung für Änderungen der Gradzahl
	`writing-mode`	Keine Unterstützung für Schreibrichtungen außer von rechts nach links	Keine Unterstützung für Schreibrichtungen, Standard ist von links nach rechts	Eingeschränkte Unterstützung, unterstützt wird nicht 'top to bottom, right to left'
`<fo:color-profile />`	`color-profile-name` `src`	Keine Unterstützung von Farbprofilen	Unterstützung von Farbprofilen	Unterstützung von Farbprofilen

`<fo:retrieve-marker />`	`retrieve-boundary`
	`retrieve-position`
	`retrieve-class-name`
		Keine Unterstützung, wenn Element von einem Block umgeben wird und die Schriftart gesetzt wird	Unterstützung bei Definition einer Schriftart über Blockelement	Unterstützung bei Definition einer Schriftart über Blockelement

`<fo:table />`	`border-top-style`	Eingeschränkte Unterstützung für Linienstil mit 'solid, none und hidden'	Unterstützung aller Linienstile	Unterstützung aller Linienstile

Element	Attribut	Apache FOP	RenderX XEP	Antenna House XSL Formatter
<fo:inline />	baseline-shift	Keine Unterstützung für Hoch- oder Tiefstellung	Unterstützung für Hoch- und Tiefstellung	Unterstützung für Hoch- und Tiefstellung
	font-family	Keine direkte Unterstützung, Konfigurationsdatei mit Verweise auf Schriftarten	Unterstützung der Schriftarten	keine Unterstützung für die Schriftart Symbol
<fo:external-graphic />	content-height content-width	Keine Unterstützung der Attribute für Höhe und Breite, Benutzung der Attribute width und height	Unterstützung der Attribute für Höhe und Breite einer Grafik	Unterstützung der Attribute für Höhe und Breite einer Grafik
		Unterstützt nicht jedes Grafikformat, bspw. .tif	Unterstützung verschiedenster Grafikformate	Unterstützung verschiedenster Grafikformate
<fo:block />	margin-left	Keine Unterstützung bei Prozentangaben	Unterstützung aller Angaben	Unterstützung aller Angaben
	word-spacing	Keine Unterstützung für größere Wortabstände	Unterstützung für Änderungen der Wortabstände	Unterstützung für Änderungen der Wortabstände
	text-transform	Keine Unterstützung für Groß- bzw. Kleinbuchstaben	Unterstützung für Groß- bzw. Kleinbuchstaben	Unterstützung für Groß- bzw. Kleinbuchstaben
	text-align-last	Keine Unterstützung für Ausrichtung in einer letzten Zeile	Unterstützung für Ausrichtung	Unterstützung für Ausrichtung
	hyphenation-character	Keine Unterstützung für Auswahl des Trennzeichens	Unterstützung für Auswahl des Trennzeichens	Unterstützung für Auswahl des Trennzeichens
	hyphenate	Keine Unterstützung für Trennung der Zeichenfolgen	Unterstützung für Trennung der Zeichenfolgen	Unterstützung für Trennung der Zeichenfolgen
<fo:leader />	rule-style	Eingeschränkte Unterstützung für die Linienstile 'dotted, dashed, solid und double'	Unterstützung für Linienstile	Unterstützung für Linienstile
<fo:float />	float	Keine Unterstützung für das Float-Konzept	Unterstützung des Float-Konzepts	Unterstützung des Float-Konzepts

Tab. 4-1 Tabelle mit Überblick über Probleme der FO-Prozessoren

4.6.2 Grafische Unterschiede

In der folgenden Tabelle sind nochmals die Unterschiede der einzelnen FO-Prozessoren zusammengetragen worden. Die Unterschiede der FO-Prozessoren beziehen sich auf die Darstellung des Seitenlayouts.

Element	Attribut	Apache FOP	RenderX XEP	Antenna House XSL Formatter
`<fo:footnote />`		Inhalt einer Fußnote wird auf gleicher Zeilenhöhe hinter Nummerierung platziert	Inhalt einer Fußnote wird etwas nach unten hin zur Nummerierung versetzt	Inhalt einer Fußnote wird etwas nach unten hin zur Nummerierung versetzt
`<fo:table-column />`	`column-width`	Tabelle wird nach Umsetzung dargestellt	Tabelle wird nach Umsetzung dargestellt	Darstellung der Tabelle weicht ab, Spalten sind verzogen, Anpassung notwendig
`<fo:list-item/>`		Inhalt eines Listenpunktes wird etwas nach unten hin der Nummerierung platziert	Inhalt eines Listenpunktes wird zur Nummerierung hin nach oben platziert	Inhalt eines Listenpunktes wird zur Nummerierung hin nach oben platziert
`<fo:block />`	`font-family`	Benötigt Konfigurationsdatei, um Schriftarten zu erkennen	Unterschiede beim Benennen der Schriftart, anstatt Courier New nur Courier	Unterstützung aller Schriftarten
`<fo:table />`	`border-top-width`	Unterstützung der Linien in einer Tabelle, nicht in den einzelnen Bereichen wie `fo:region-body`	Darstellung der Linien sind flacher als bei der Formatierung des Apache FOP	Darstellung der Linien sind dicker als bei der Formatierung des RenderX XEP

Tab. 4-2 Tabelle mit Überblick über Unterschiede der FO-Prozessoren

5 Anforderungen an die XSL-FO Entwicklungsumgebung

Dem Anwender soll eine XSL-FO Entwicklungsumgebung zur Verfügung gestellt werden, über die es ihm möglich ist, XSL-FO Projekte anzulegen und zu verwalten. Diese Projekte erlauben es, FO-Dokumente dynamisch zu erzeugen und zu bearbeiten. Durch die Publikation dieser FO-Dokumente ist dem Anwender das Erstellen von PDF-Dokumenten über einen, ihm zur Verfügung gestellten, FO-Prozessor zugänglich.

Der Anwender besitzt dabei ein XML-Dokument, welches die Inhalte bspw. seiner Technischen Dokumentation strukturiert anbietet. Über XSL-FO-Stylesheets werden diese Inhalte erfasst und nach FO übertragen. Standardtemplates, die den Inhalt des Druckbereichs auf die Seiten bringen, sollen durch einen FCT-Konfigurator bereitgestellt werden. Ein Power-User soll Inhaltstemplates aus diesen Standardtemplates ableiten, um diese anpassen zu können. Die Inhaltstemplates, die durch den Power-User bereitgestellt werden, können dann vom Anwender genutzt werden, um den entsprechenden Inhalt auf den Seiten zu platzieren. Das FO-Dokument sollte mit Hilfe des FO-Prozessors, in diesem Fall entweder Apache FOP, Antenna House XSL Formatter oder RenderX XEP, in ein PDF-Dokument überführt werden können.

Neben dem Anlegen und Bearbeiten solcher Projekte sollte es dann möglich sein, die Konfigurationen hierfür auf einen anderen Server zu übertragen. Das hat den Vorteil, dass über diese Konfiguration nun einfach und bequem PDF-Dokumente erzeugt werden können, durch Ausführung gewisser Batchprozesse. Diese Prozesse verarbeiten dann das vom Anwender angebotene XML-Dokument zusammen mit dem von der Anwendung erstellte XSL-FO Stylesheet.

Bisher hat FCT ein komplettes XSL-FO Stylesheet, welches die Definition von Master-Pages sowie Seitenfolgen und deren Inhalte beschreibt. Das hat den Nachteil, dass diese Einstellungen in einem kompletten Stylesheet zusammengetragen werden müssen. Dies betrifft die Gestaltung der einzelnen Seitenvorlagen und Seitenfolgen. Dieses Stylesheet wird von der Firma FCT dem Kunden als Standard mitgeliefert und muss geringfügig für den Kunden angepasst werden.

Die Anforderungen an dieses Stylesheet kommen vom Anwender und werden von den Entwicklern der Firma FCT umgesetzt. Diese Anforderungen betreffen hauptsächlich das Seitenlayout. Sollten sich nachträglich Änderungen am Seitenlayout ergeben, so muss der Weg wieder über die Firma gehen, ohne dass der Anwender dies direkt selbst vornehmen kann. Dieser Weg hin bis zur Erstellung des Ergebnisdokuments, soll durch diese Diplomarbeit auf drastische Weise vereinfacht werden. Der Anwender selbst kann gezielt über die Anwendung angeben, wie die einzelnen Seiten auszusehen haben und welche Inhalte auf den Seiten platziert werden sollen. Die Darstellung des Seiteninhalts, betreffend des Druckbereichs, kann er anhand des XSL-FO Stylesheets, welches von FCT bereitgestellt wurde, direkt auswählen.

Um ein komplettes XSL-FO Stylesheet zu entwickeln, bedarf es eines großen Zeitintervalls, da die Entwicklung zeitintensiv und aufwendig ist. Dies hängt von der Anzahl der verschiedenen Elemente des XML-Dokuments ab, die in XSL-FO umgesetzt werden sollen. Eine weitere Schwierigkeit taucht auf, wenn bestimmte Veränderungen am PDF-Dokument vorgenommen werden sollten. Diese Veränderungen betreffen Einstellungen bezüglich der Seitenränder, des Papierformat oder der Absatzgestaltung. Es können ebenso Änderungen am Inhalt der Seiten vorgenommen werden, die die weitere Handhabung des Stylesheets erschweren. Sollte eine Änderung vorgenommen werden, kann das Stylesheet über einen entsprechenden XSL-FO Editor bearbeitet werden. Die Parameter, die geändert werden sollten, müssen über die Suchfunktion aufgespürt werden. Wenn die Stelle dann gefunden wurde, wird der Wert geändert und der FO-Pro-

zessor erneut aufgerufen. Dabei sinkt die Benutzerfreundlichkeit enorm. Meistens sind es nicht nur Veränderungen an einer Stelle, sondern an vielen Stellen in diesem Stylesheet.

Um das komplette Dokument im Überblick zu behalten, müssen XSL-FO Kenntnisse vorausgesetzt werden, um den Sinn und die Bedeutung der einzelnen Elemente zu verstehen. Sollten nachtragend neue Elemente in der Struktur des XML-Dokuments eingeführt werden, so müssen diese Änderungen durch die Firma FCT angepasst werden. Anhand dieser Gesichtspunkte und Problematiken, lassen sich folgende Anforderungen an die XSL-FO Entwicklungsumgebung stellen:

Der Anwender soll die XSL-FO Entwicklungsumgebung bei sich auf dem Arbeitsplatz installieren und konfigurieren können. Nachdem er diese Umgebung installiert hat, soll es ihm möglich sein, XSL-FO Projekte anlegen zu können. Zu diesen Projekten soll der Anwender die vorhandenen XSL-FO Prozessoren und die verwendeten FO-Stylesheets angeben können. Die durch den Anwender angelegten XSL-FO-Projekte sollten dann wiederum geladen und bearbeitet werden können. Dem Anwender soll es nun möglich sein, Seitenvorlagen (im Kommenden mit Master-Pages bezeichnet), Seitensequenzen zur Bearbeitung statischer Inhalte, sowie die eigentlichen Seiteninhalte und Absatzformate anlegen und bearbeiten zu können. Durch die Angabe dieser Einstellungsmöglichkeiten soll dann ein PDF-Dokument generiert werden können. Nachdem ein Projekt konfiguriert wurde, soll es die Möglichkeit geben, es auf anderen Servern lauffähig halten zu können, um so PDF-Dokumente über die zuvor eingestellten Konfigurationen ausspielen zu können. Die folgende Abbildung verdeutlicht die Anforderungen an das System.

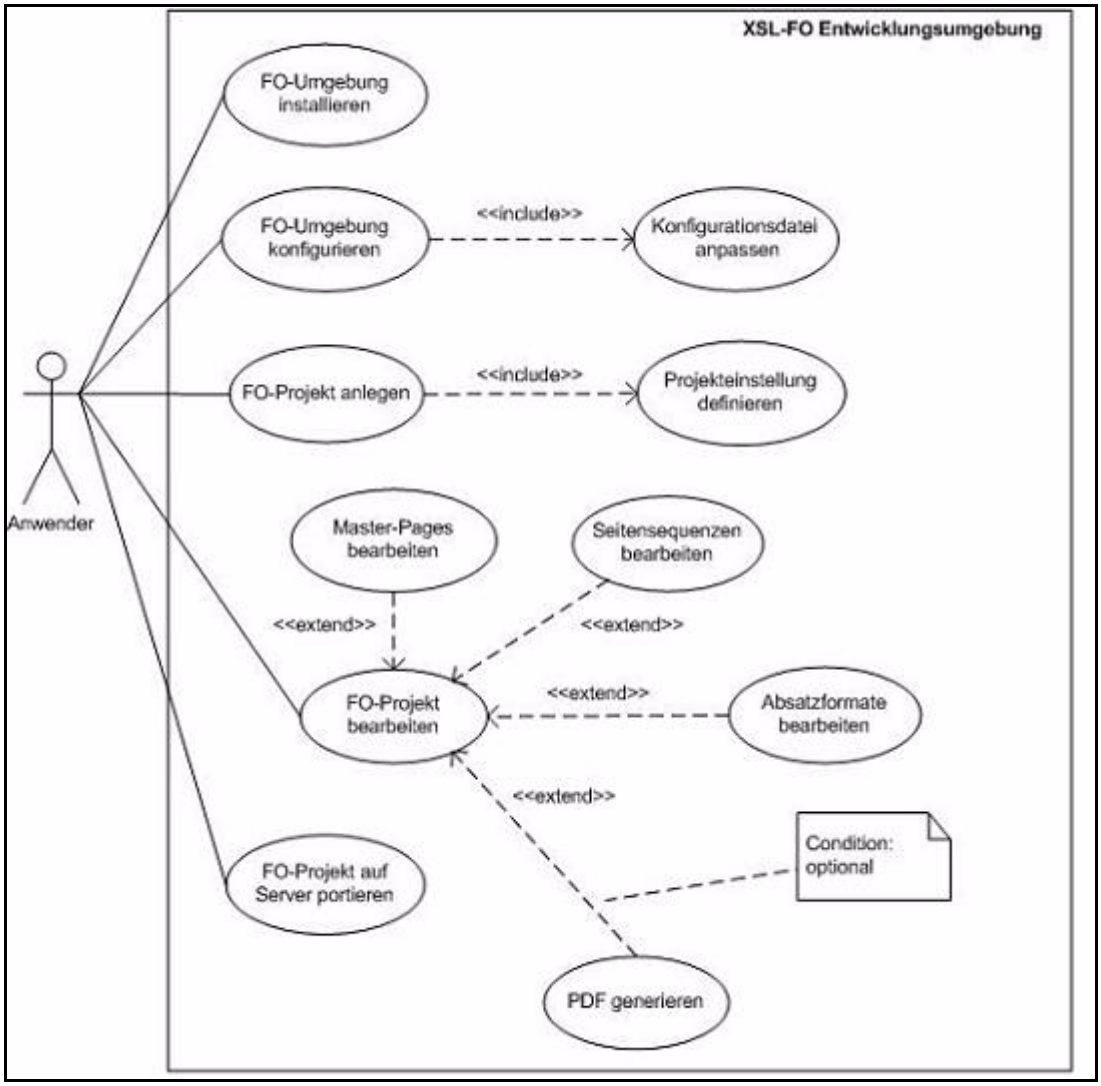


Fig. 5-1 : Use-Case Diagramm, Anforderungen Entwicklungsumgebung

Damit der Anwender Inhalte auf die jeweiligen Druckbereiche einer Seite platzieren kann, muss ein XSL-FO Stylesheet zur Verfügung gestellt werden. Die folgende Abbildung verdeutlicht die Anforderungen.

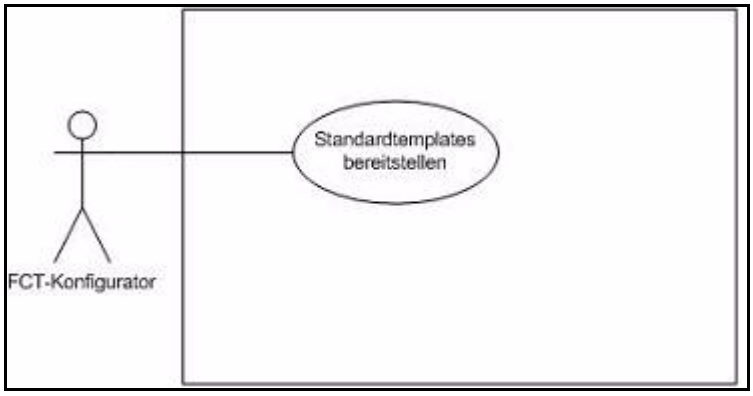


Fig. 5-2 : Use-Case Diagramm, FCT-Konfigurator

Diese Standardtemplates treffen auf Änderungswünsche des Kunden. Der Power-User leitet hierzu aus diesen Standardtemplates Inhaltstemplates ab und passt diese entsprechend an. In diesen Templates werden die Strukturelemente

des Kundendokuments verarbeitet und FO-Elemente gesetzt, die die Inhalte des XML-Dokuments ausspielen. Diese Inhaltstemplates werden dann durch den Power-User dem Anwender bereitgestellt. Die folgende Abbildung soll auch diese Anforderung verdeutlichen.

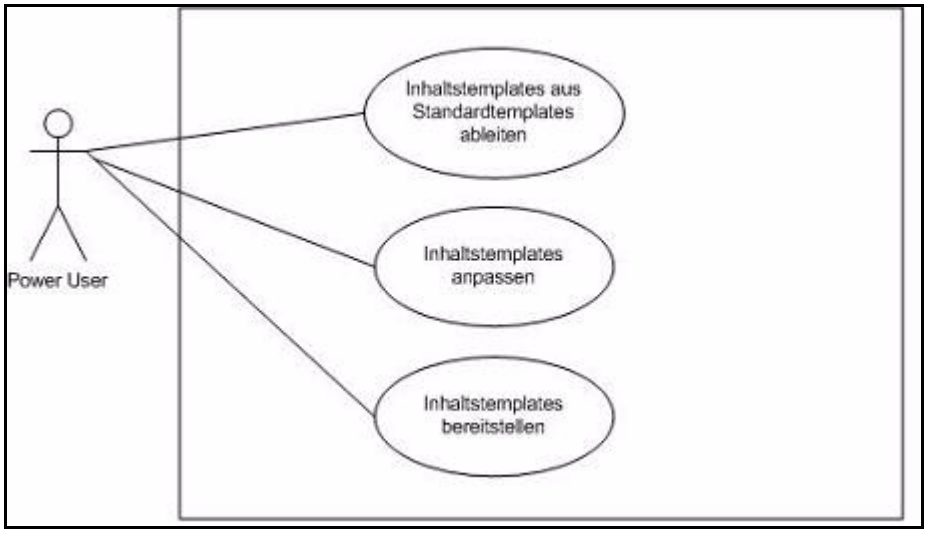


Fig. 5-3 : Use-Case Diagramm, Power-User

Die folgenden Kapitel beschreiben die einzelnen Use-Cases.

5.1 FO-Entwicklungsumgebung installieren

Bevor der Anwender die XSL-FO Entwicklungsumgebung verwenden kann, muss er diese in sein System integrieren. Die Integration sollte anhand einer Installationsanleitung oder einer ausführbaren Datei die sämtliche Dateien des Systems in ein Verzeichnis kopiert und gegebenenfalls die Pfade in der Konfigurationsdatei anpasst erfolgen. Durch die Installation der XSL-FO Entwicklungsumgebung stehen die Standardtemplates zur Verfügung.

Eine Installationsanleitung wäre recht einfach und erfordert weniger Zeitaufwand. In dieser Anleitung wird dem Anwender Schritt für Schritt erklärt, welche Verzeichnisse und Dateien an welche Stellen des Systems kopiert werden müssen, damit die Anwendung einwandfrei funktionieren kann. Ebenso wird ihm erklärt, welche Konfigurationseinträge er ändern muss, um seine auf dem System installierten FO-Prozessoren der Anwendung bekannt zu geben. Die andere Möglichkeit wäre eine ausführbare Setup-Datei, die den Anwender durch die einzelnen Installationsschritte leitet. Dies wäre recht aufwendig.

Nach der Installation sollte die Anwendung ohne Probleme vom Anwender ausgeführt werden können. Hilfreich für den Anwender ist dabei eine Dokumentation der Entwicklungsumgebung, in der Schritt für Schritt die einzelnen Funktionen und Menüpunkte der Anwendung beschrieben werden. Somit sollte dem Anwender das Arbeiten mit der Entwicklungsumgebung leicht gemacht weden, damit er sich sofort zurechtfindet und er somit gleichzeitig bei seinen Konfigurationen unterstützt wird.

5.2 FO-Entwicklungsumgebung konfigurieren

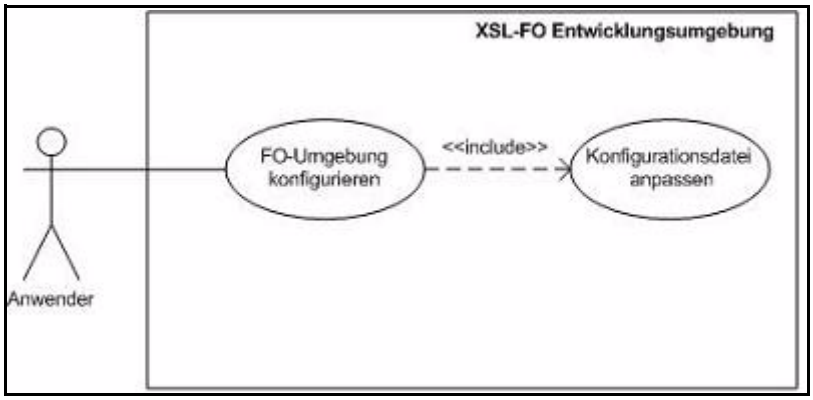


Fig. 5-4 : Use-Case Diagramm, Konfiguration Entwicklungsumgebung

Wenn der Anwender die Anwendung das erste Mal ausführt, sollte er wichtige Einstellungen über eine Konfigurationsdatei definieren können. Dazu zählen FO-Prozessoren, die verwendet werden sollen, sowie XSLT-Prozessoren. Eine weitere wichtige Einstellung ist die Angabe der einzelnen Stylesheets, die zur Erzeugung des FO-Dokuments dienen.

Desweiteren sollten Verzeichnisse angegeben werden, wo die generierten Batch-Dateien und XSLT-Stylesheets liegen sollen, damit diese im weiteren Verlauf über einen weiteren Server zur Verfügung gestellt werden können. Dies hat den Vorteil, dass sämtliche Einstellungen vom Anwender in diesen Stylesheets und XML-Dokumenten gespeichert worden sind und er nur noch die Batch-Dateien auszuführen braucht, damit seine FO-Dokumente, bzw. PDF-Dokumente erstellt werden können.

Konfigurationsdatei anpassen

Die Konfigurationsdatei soll so angepasst werden, dass der Anwender zwischen den zur Verfügung stehenden Prozessoren wählen kann. Damit die Anwendung diese Prozessoren erkennt, muss der Pfad in einer Variablen gespeichert werden. Außerdem sollte der Name des Prozessors konfigurierbar sein, damit dieser auch in der Anwendung über eine Auswahlliste angesprochen werden kann.

Jeder FO-Prozessor hat Unterschiede, was das Aufrufen der Parameter über die Kommandozeile betrifft. Die folgende Abbildung zeigt diese Unterschiede für die FO-Prozessoren Apache FOP, Antenna House XSL Formatter und RenderX XEP.

```
'1. Apache Fop '
'2. Antenna House Formatter '
'3. RenderX XEP '

fop.bat -c userconfig.xml
        -fo OBJ_20070919.fo
        -pdf OBJ_20070919.pdf

XSLCmd.exe -d OBJ_20070919.fo
           -o OBJ_20070919.pdf

xep.bat -fo OBJ_20070919.fo
        -pdf OBJ_20070919.pdf
```

Fig. 5-5 : Kommandozeilenaufrufe mit FO-Prozessoren

Apache FOP benötigt folgende Übergabeparameter in der Konfigurationsdatei: Den Parameter *-c,* um eine Konfigurationsdatei mit anzugeben, über welche die unterschiedlich im FO-Dokument verwendeten Schriftarten bezogen werden können. Wenn dieses nicht mit angegeben wäre, so würde der FO-Prozessor manche Schriftarten nicht kennen und somit auch im Ergebnisdokument nicht richtig darstellen können. Weitere Parameter, die der Prozessor benötigt sind *-fo* für das jeweilige FO-Dokument, welches formatiert werden soll, sowie *-pdf* für die Umwandlung in das Format PDF.

Völlig andere Übergabeparameter benötigt der FO-Prozessor Antenna House XSL Formatter. Er benötigt keine Konfigurationsdatei, um die Schriftarten zu unterstützen. Jegliche Schriftarten, die mit dem Formatter unterstützt werden, können über das Optionsmenü des Formatters nachgeschaut werden. Damit der Formatter weiß, welches FO-Dokument formatiert werden soll, muss der Parameter *-d* angegeben werden. Damit das Ergebnisdokument erstellt werden kann, muss der Parameter *-o* mit dem Dateinamen angegeben werden.

Der FO-Prozessor RenderX XEP benötigt wie der Apache FOP die beiden Parameter *-fo* für das FO-Dokument und *-pdf* für die Erzeugung des Ergebnisdokuments. Dabei kann es für jeden Parameter eine eigene Variable geben, die den Aufrufparameter beinhaltet. Somit wüsste die Anwendung genau, wie die einzelnen Parameter heißen. Zum anderen gibt es die Möglichkeit, nur eine Variable hierfür zu verwerden. Das hat den Vorteil, dass das Ganze übersichtlicher bleibt und dem Anwender sofort klar wird, welche Werte hier mit angegeben werden müssen.

Jedoch benötigen alle Prozessoren eine optionale Variable für die Konfigurationsdatei der unterschiedlichen Schriftarten. Dies ist nötig, da der Apache FOP nur so manche Schriftarten richtig unterstützen kann. In dieser Variablen kann der Anwender dann den Pfad zur Konfigurationsdatei mit angeben.

Weitere Einstellungen betreffen, wie bereits erwähnt, Pfade für die generierten Dateien. Es werden Stylesheets und Batchdateien erzeugt, die in Verzeichnisse abgelegt werden müssen. Die Verzeichnisse sollte der Anwender ebenso in dieser Konfigurationsdatei mit angeben können. Des Weiteren sollte der Anwender einen XSLT-Prozessor angeben können, durch den das XML-FO Dokument erzeugt wird. Hierzu muss die Konfigurationsdatei eine Variable beinhalten, welche diesen XSLT-Prozessor beinhaltet. Die Konfiguration entspricht genau der der FO-Prozessoren. Diese Anforderung soll hier nicht weiter vertieft werden.

5.3 FO-Projekt anlegen

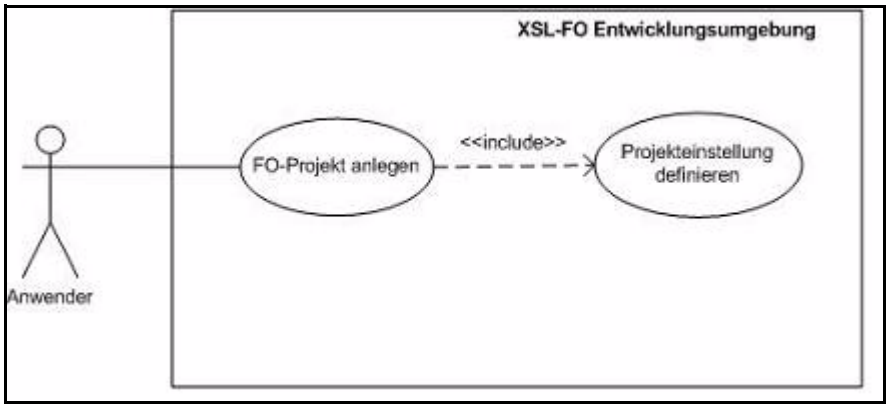


Fig. 5-6 : Use-Case Diagramm, neues FO-Projekt anlegen

Der Anwender sollte über die Entwicklungsumgebung XSL-FO Projekte anlegen können. Projekte können unterschiedliche Konfigurationen besitzen. Wenn der Anwender ein neues XSL-FO Projekt anlegt, dann sollten gewisse Projekteinstellungen definiert werden können, die beim Öffnen des Projekts geladen werden sollten. Parameter, die dabei vergeben werden, sind vor allem der Name des Projekts, der es erlaubt, ein Projekt eindeutig zu identifizieren, der Pfad zur im vorherigen Kapitel erwähnten Konfigurationsdatei, welche Einstellungen der XSL-FO und XSLT-Prozessoren beinhaltet, sowie die Speicherorte für die Konfigurationen der Seitenvorlagen (Master-Pages), Seitensequenzen und Absatzformate. Diese werden noch in den weiteren Kapitel dieser Diplomarbeit näher beschrieben.

Die Einstellungen, die beim Anlegen des Projekts angegeben werden, sollten dann im System des Anwenders gespeichert werden. Beim Öffnen eines Projekts werden die Daten ausgelesen und in die Entwicklungsumgebung geladen. Somit arbeitet der Anwender immer nur auf dem aktuellen Projekt. Nachdem der Anwender ein FO-Projekt angelegt hat, sollten nun zusätzliche Einstellungen getroffen werden können, um das PDF-Dokument zu generieren. Hierzu zählen Angaben zum verwendeten FO-Prozessors, zum Speicherort für erzeugte FO-Dokument sowie der Angabe des XSL-FO Stylesheets, welches durch den Power-User angepasst und bereitgestellt wurde.

Damit dieses XSL-FO Stylesheet auch verwendet werden kann, muss das XML-Dokument zusätzlich mit angegeben werden können, damit die entsprechenden Inhalte der Dokumentationen über die Inhaltstemplates des Stylesheets nach FO überführt werden können. Nachdem der Anwender nun die Projekteinstellungen definiert hat, kann er das eigentliche FO-Projekt bearbeiten. Diese Anforderungen werden im folgenden Kapitel näher beschrieben.

5.4 FO-Projekt bearbeiten

Nachdem der Anwender nun die Einstellungen für sein Projekt konfiguriert hat, kann er auf die Gestaltung des Seitenlayouts sowie auf die Inhalte der einzelnen Seite eingehen. Die folgende Abbildung zeigt die Anforderungen zum Use-Case *FO-Projekt bearbeiten*.

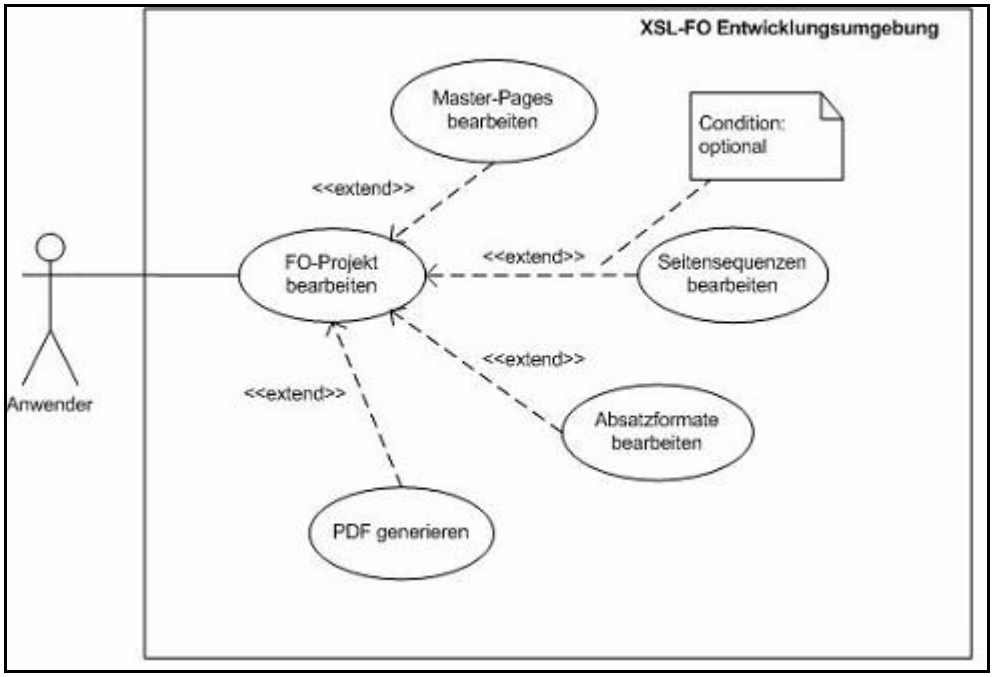


Fig. 5-7 : Use-Case Diagramm, FO-Projekt bearbeiten

Beim Bearbeiten eines FO-Projekts sollen Master-Pages, Seitensequenzen und Absatzformate konfiguriert werden können. Nach dem Anlegen einer Master-Page und einer Seitensequenz soll der Anwender ein PDF-Dokument generieren lassen können. Wenn hier von der Bearbeitung von Master-Pages gesprochen wird, dann ist damit das Bearbeiten einer Seitenvorlage gemeint. Zur besseren Vorstellung kann man diesen Prozess mit Microsoft PowerPoint vergleichen. Hier können sogenannte Master angelegt werden, die dann den einzelnen Folien zugewiesen werden können. In XSL-FO ist das genauso. Es können Master-Pages angelegt werden, die dann den Seitensequenzen zugewiesen werden können. Somit bekommt die Seitensequenz, wie die Folie, einen Master zugewiesen und nimmt somit dessen Seitenlayout an.

Bei der Bearbeitung der Seitensequenzen geht es darum, statische Inhalte auf eine Seite zu bringen. Wenn der Anwender eine Master-Page angelegt hat, dann kann er diese für eine Seitenfolge (Seitensequenzen) referenzieren und verschiedene Inhalte in Kopf- und Fußzeile sowie linkem und rechtem Seitenbereich zuweisen. Bei diesen Einstellungsmöglichkeiten sollte es nun möglich sein, Inhaltstemplates auszuwählen bzw. referenzieren zu können, um für den Druckbereich einer Seite den Inhalt vergeben zu können. Hierzu soll eine Lösungsmöglichkeit gefunden werden, die diesen Schritt möglichst einfach und vor allem für den Anwender benutzerfreundlich umsetzt.

Um Einfluss auf die verschiedenen Formatierungen innerhalb dieser Bereiche nehmen zu können, sollte die Entwicklungsumgebung die Möglichkeit bieten, verschiedene Absatzformate definieren zu können. Um den Prozess des Bearbeitens eines XSL-FO Projekts näher betrachten zu können, wird ein sogenanntes *Getting Started* betrachtet, dass in einem Überblick den Prozess vom Anlegen eines FO-Projekts bis hin zum fertigen PDF-Dokument betrachten soll.

Getting Started

Der erste Schritt ist, ein neues XSL-FO Projekt anzulegen. Dabei gibt der Anwender die benötigten Konfigurationseinstellungen an. Nehmen wir das Beispiel eines Deckblattes, um zu zeigen, wie der Anwender einfach und bequem zu einer PDF-Ausgabe gelangt.

Das erste was benötigt wird, ist eine Master-Page, die das Seitenlayout des Deckblatts darstellen soll. Seitliche Abstände, das Seitenformat sowie Einstellungen für Kopf- und Fußzeile, linker und rechter Seitenbereich als auch den Druckbereich können hier eingestellt werden. Die Möglichkeiten, die XSL-FO bietet, sollen durch eine entsprechende Oberfläche dem Anwender zur Verfügung gestellt werden können. Master-Pages sind von Bedeutung, da diese dann über die Seitensequenzen referenziert werde.

Nachdem die Master-Page angelegt wurde, soll nun das eigentliche Deckblatt erstellt werden. Das bedeutet, man benötigt eine Seitensequenz, die in diesem Fall aus einer Seite besteht. Hier soll nun eine entsprechende Seitenvorlage ausgewählt werden können. Durch die Definition dieser Seitenvorlage können die einzelnen Bereiche bearbeitet werden. Der Anwender kann somit statische Inhalte vergeben, z.B. Grafiken, Kapitelüberschriften, Seitenzahlen oder Zeichenfolgen. Neben den statischen Inhalten gibt es den Inhalt für den Druckbereich, der die eigentliche Information einer Seite bereitstellt. Dieser Inhalt kommt aus dem XML-Dokument des Anwenders. Dieses wurde bereits bei den Projekteinstellungen mit angegeben. Die Verarbeitung dieser Inhalte nach FO übernimmt ein XSL-FO Stylesheet. Dieses Stylesheet wird durch den FCT-Konfigurator bereitgestellt und durch den Power-User kundenspezifisch angepasst. Die Inhaltstemplates dieses Stylesheets sollen nun über die Anwendung ausgewählt werden können, um so zu bestimmen, welche Inhalte auf die Seiten gesetzt werden sollen.

Nachdem die Seitensequenz nun angelegt wurde, soll der Anwender die Möglichkeit besitzen, Absatzformate definieren zu können. Diese Absatzformate können dann in den Bereichen einer Seite verwendet werden. Das Absatzformat kann verändert und über die Generierung des PDFs die Auswirkungen betrachten werden. Das erstellte Absatzformat sollte auch den Bereichen mit statischen Inhalten zugewiesen werden können. Um jedoch diese Absatzformate auch für den eigentlichen Inhalt nutzen zu können, muss der Power-User diese in den Inhaltstemplates anlegen.

Nachdem der Anwender auch diesen Schritt erledigt hat, kann er nun das PDF-Dokument generieren lassen. Wenn alle Einstellungen hierzu angegeben worden sind, so sollte über den eingestellten FO-Prozessor die Ausgabe nach PDF erfolgen und das Deckblatt mit dessen Seiteninhalt über den Adobe Reader dargestellt werden. Die folgende Abbildung zeigt den gesamten Prozess nochmals im Überblick.

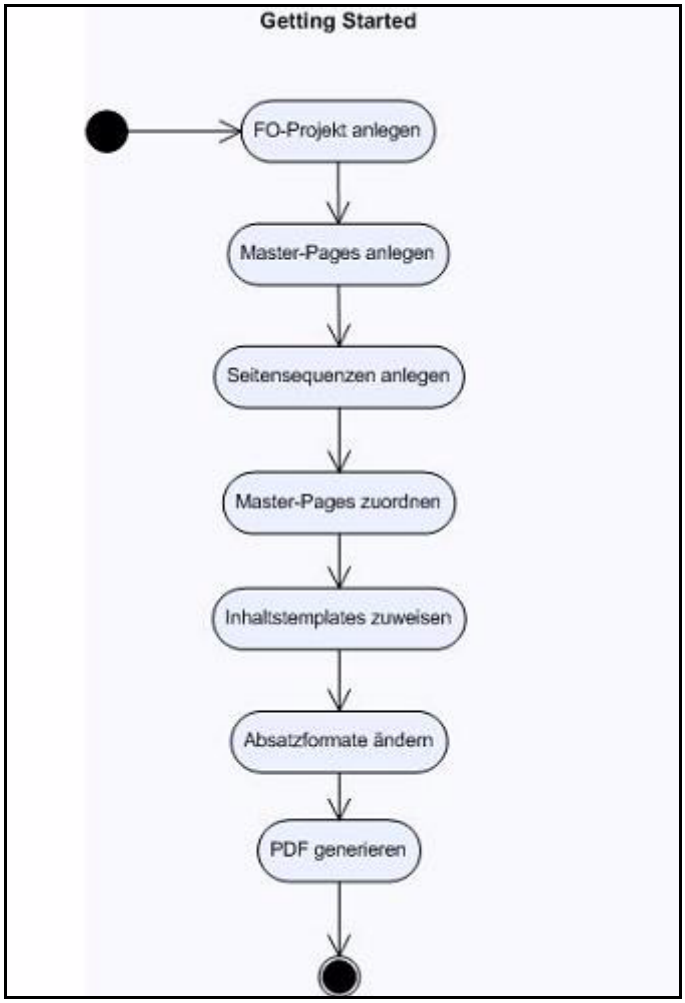


Fig. 5-8 : Ablaufdiagramm, XSL-FO Projekt anlegen

Nachdem die Konfigurationen für die Generierung des PDF-Dokuments gespeichert wurden, kann der Anwender nun diese Einstellungen auf andere Server übertragen, um diese von dort aus auszuführen. Durch die FO- sowie XSLT-Prozessoren werden Batchprozesse angestoßen, die jeweils das FO- und das PDF-Dokument generieren. Solche Batchprozesse müssen nur einmal aufgerufen werden um zum Ergebnisdokument zu gelangen. Die Gestaltung des Seitenlayouts sowie Zuweisungen von Inhalten und Absatzformaten, soll über die Anwendung erfolgen. Nachdem dann diese Angaben gespeichert worden sind, liegt zu diesem Zeitpunkt das Layout der Seiten fest und braucht somit nicht wieder über die Anwendung angestoßen werden. Durch Batchprozesse wäre es nun möglich, die Ausspielung der PDF-Dokumente ohne die Anwendung zu erzwingen. Diese Prozesse könnten selbständig auf anderen Servern laufen und von dort aus angesprochen werden. Die restlichen Anforderungen, die beim Bearbeiten eines Projekts berücksichtigt werden sollten, sind in den nachfolgenden Kapitel näher beschrieben.

5.4.1 Master-Pages bearbeiten

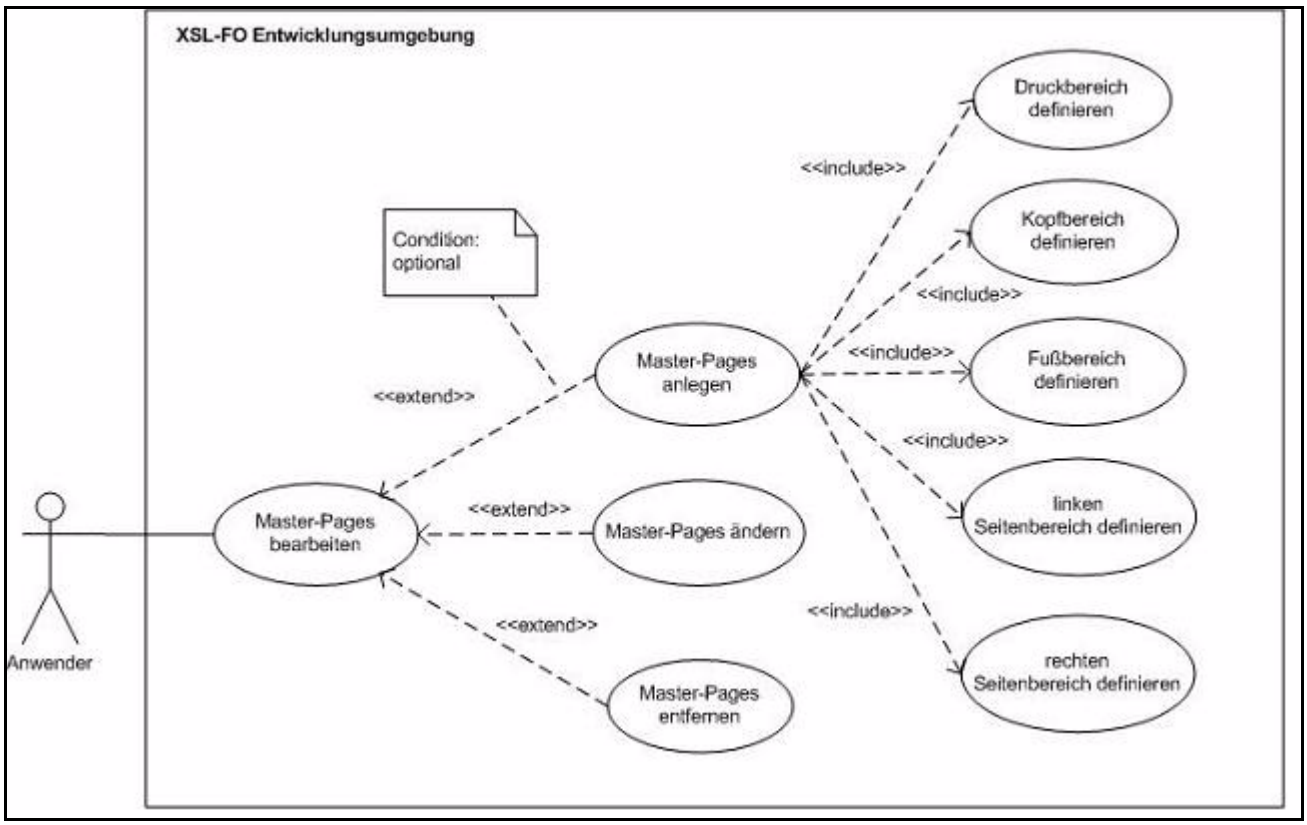


Fig. 5-9 : Use-Case Diagramm, Master-Pages bearbeiten

Das Bearbeiten einer Master-Page beinhaltet die Use-Cases Master-Pages anlegen, Master-Pages ändern und Master-Pages entfernen. Die Anforderungen hierzu werden in den folgenden Abschnitten näher beschrieben.

Master-Pages anlegen

Das Anlegen einer Master-Page gehört wohl zum wichtigsten Prozess in diesem Ablauf. Der Name einer Master-Page sollte beim Erstellen eindeutig vergeben werden, da über diesen Namen eine Zuordnung zu den jeweiligen Seitensequenzen referenziert werden kann. Somit kann der Anwender jegliche Master-Pages für die Seitensequenzen austauschen um zu sehen, wie sich diese auf das Seitenlayout auswirken. Über den Namen kann eine Master-Page leicht gefunden und bearbeitet werden.

Desweiteren soll ein geeignetes Seitenformat vergeben werden können. Hier könnte es für den Anwender zwei Möglichkeiten geben, wie er diese über die Anwendung eingeben könnte.

Zum einen könnte es eine Auswahl an DIN-Formaten der Form A0, A1, A2, usw. geben, wobei sich hier das jeweilige Format auswählen lässt. Zusätzlich könnten Einstellungen wie Hoch- und Querformat definiert werden. Dies hätte den entscheidenden Nachteil, dass die Größen durch die Anwendung vorgegeben wären. Das bedeutet, der Anwender ist in dieser Einstellung eingeschränkt und könnte somit z.B. keine breiteren Seiten definieren. Deshalb wäre es geschickter, wenn der Anwender diese Höhen- und Breitenangaben selbst tätigen kann. Somit wäre zusätzlich zur Auswahl von DIN-Formaten die Einstellung *Benutzerdefiniert* für einen Anwender sehr wichtig. Er könnte fehlende Formate definieren

und diese den Vorlagen zuordnen. Anwendungsbeispiele in Technischen Dokumentationen sind Ausklappseiten, um Grafiken besser darstellen zu können.

XSL-FO bietet noch weitere Möglichkeiten, den Bereich einer Seiten weiter einzuschränken. Es ist möglich Seiten in ihren seitlichen Abständen einzuengen. Man kann sich ein Viereck vorstellen, das man nach allen Seiten hin in seiner Größe verziehen kann. Dies bedeutet, dass man Marginalien schaffen kann, um so den Seiten ein noch besseres Design zu verschaffen. Wie diese Einschübe zu verstehen sind, soll die nachfolgende Abbildung verdeutlichen.

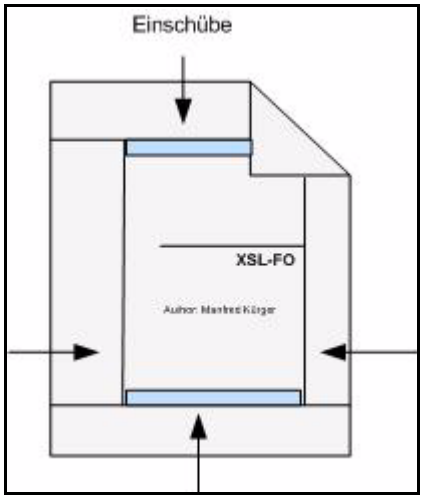


Fig. 5-10 : Einschübe einer Seite

Durch die seitlichen Einschübe ist es möglich, Ränder zu erzeugen. Diese Einstellungen haben noch nichts mit den Inhalten der eigentlichen Seite zu tun. Sie tragen wesentlich der Erzeugung des Seitenlayouts bei. Diese Einstellungen sind ein großer Vorteil für den Anwender als auch für den Entwickler, da diese Einstellungen (Master-Pages) nun nicht mehr in dem XSL-FO Stylesheet definiert werden müssen. Der Anwender kann so flexibel und ohne großen Zeitaufwand jegliche Änderungen am Design seiner Seiten vornehmen, um das gewünschte Aussehen damit zu erreichen. Nun ist es ja nicht immer so, dass es für jede Seite eine Master-Page gibt, sondern sich diese auf mehrere Seiten bezieht. Wenn man sich z.B. ein Inhaltsverzeichnis anschaut, so wird man feststellen, dass es sich nicht nur auf eine Seite beschränkt. Bei dieser Anforderung soll dem Anwender die Möglichkeit eingeräumt werden, zu bestimmen, ob sich die Seitenvorlage über eine oder mehrere Seiten hinweg erstreckt.

Wenn diese Grundlage geschaffen ist, kann der Seiteninhalt bearbeitet werden. Dieser Inhalt lässt sich aufteilen in den eigentlichen Druckbereich, die Kopfzeile, Fußzeile, linker und rechter Seitenbereich.

Druckbereich definieren

Der Druckbereich der Seite sollte sich ebenso weiter eingrenzen lassen wie die Master-Page auch. Diese Einstellungen sollten über die Anwendung getroffen werden können, um die Flexibilität des Prozesses weiter zu erhöhen. Die Definitionen sollten sich wieder auf den oberen, unteren, linken und rechten Seitenbereiche des Druckbereichs beziehen. Der Anwender sollte die Möglichkeit haben, diese Werte selbst einzugeben, ohne diese aus einer Liste auszuwählen. Bei solchen Auswahlmöglichkeiten ist der Benutzer stark in der Wahl eingeschränkt. Er hätte somit nicht viel Bewegungsfreiheit in seinen Vorstellungen und wäre somit von der Anwendung abhängig. Neben diesen einstellbaren Parametern bietet XSL-FO weitere Möglichkeiten bezüglich des Seitenlayouts. Nicht jedes Dokument besitzt einen transparenten Hintergrund. Der Druckbereich kann eine Hintergrundfarbe besitzen. Die Anforderungen sind in XSL-FO möglich und können durch die getesteten XSL-FO Prozessoren ohne Probleme umgesetzt werden.

Somit sollte der Anwender auch diese Eigenschaft über die Anwendung setzen können. Weitere Möglichkeiten die XSL-FO bietet, sind Linien für Kopf- und Fußzeilen. Die untere Abbildung zeigt ein Beispiel hierfür.

Fig. 5-11 : Linien in den Kopfzeilen

Solche Einstellungsmöglichkeiten sind in diesem Fall hier fehl am Platz, da diese nicht von allen FO-Prozessoren unterstützt werden. Das Ziel dieser Diplomarbeit und somit auch der XSL-FO Entwicklungsumgebung ist es prozessorunabhängig zu bleiben, sprich mit jedem FO-Prozessor sollte ungefähr dasselbe Erscheinungsbild des PDF-Dokuments erreicht werden. Um dieser Problematik auszuweichen, sollten solche Einstellungen erst beim Definieren der Seitensequenzen zum Tragen kommen, welche im weiteren Verlauf näher geschildert werden.

Kopfbereich definieren

Um einen Kopfbereich bei späteren Definitionen referenzieren zu können, bedarf es eines eindeutigen Namens. Dies ist sehr wichtig, da sich das Layout bei geraden/ungeraden Seiten unterscheiden kann. Somit können jeweils diese Bereiche separat angesprochen werden, um diesen unterschiedliches Aussehen zu verleihen. Bei der Bearbeitung einer Seitensequenz werden diese Bereiche referenziert, um ihnen ihre Inhalte zu geben. Neben dem Namen sollte der Anwender die Ausrichtung des Kopfbereichs selbst festlegen können. Das bedeutet, dass der Abstand nach oben hin angegeben werden kann, um so nochmals einen Abstand zu erzwingen. Das hat den Vorteil, dass die Kopfzeile vertikal nach oben oder unten hin verschoben werden kann und der Anwender so nicht auf eine feste Position dieses Bereichs eingeschränkt ist.

Wie auch für den Druckbereich sollte der Anwender eine dementsprechende Hintergrundfarbe auswählen können. XSL-FO bietet noch weitere einstellbare Parameter, die jedoch nicht von allen FO-Prozessoren unterstütz werden. Deshalb sollte man sich nur hier rauf beschränken um weitere Probleme zu vermeiden.

Fußbereich definieren

Auch der Fußbereich sollte für späteres Hinzufügen von Inhalten durch einen Namen referenzierbar sein. Der Anwender sollte diesen Bereich ebenso wie den Kopfbereich vertikal nach oben oder unten hin verschieben können. Wenn jedoch keine Fußzeile benötigt werden sollten, so kann der Anwender dies über eine Definition des Wertes *0mm* erreichen. Es sollte dabei darauf geachtet werden, dass gewisse Werte bereits in der Anwendung eingestellt sind, damit der Anwender sich nicht mit den vielen Eingaben überfordert fühlt, um so bei Nichtverwendung der Einstellungen die jeweiligen Werte setzten zu müssen. Wie im Kopfbereich auch sollte die Hintergrundfarbe eingestellt werden können.

Linken Seitenbereich definieren

Der linke Seitenbereich kann bspw. für eine Marginalie genutzt werden. So wie bei Kopf- und Fußzeile sollte der Anwender hier einen Namen definieren können um beim Anlegen einer Seitensequenz diesem Bereich einen Inhalt zuzuweisen. Diesmal erstreckt sich der Abstand nach links hin horizontal und kann somit auch nach rechts verschoben werden. Auch für diese Konfiguration sollte eine entsprechende Wahl der Hintergrundfarbe getroffen werden können.

Rechten Seitenbereich definieren

Der rechte Seitenbereich kann auch als Marginalie genutzt werden und sollte über einen Namen referenzierbar sein. Dieser Bereich erstreckt sich von rechts her und kann somit horizontal nach links als auch nach rechts hin verschoben werden. Zur Darstellung des rechten Seitenbereichs gehört ebenso die Auswahl einer Hintergrundfarbe. Hierbei ist darauf zu achten, dass es nicht nur vorgegebene Hintergrundfarben geben sollte. Neben der Auswahl einer Farbpalette sollte es möglich sein, selbst Farben definieren und für diese Konfigurationen speichern zu können.

Master-Pages ändern

Nachdem eine Master-Page angelegt wurde, sollte es dem Anwender möglich gemacht werden, die Einstellung die er hier getroffen hat, zu verändern. Hier spiegelt sich nun der Vorteil dieser zu implementierenden Anwendung wieder. Über eine Oberfläche soll es möglich gemacht werden, gewisse Parameter schnell und einfach für die definierten Master-Pages ändern zu können. Das Ändern gewisser Parameter über die Oberfläche soll nun den Weg erleichtern, was zuvor über die direkte Anpassung im XSL-FO Stylesheet ausgeführt werden musste. Somit werden die Änderungen durch die Anwendung in den entsprechenden Stylesheets angepasst.

Um eine Master-Page ändern zu können, sollten dem Anwender jegliche zum aktuellen Projekt gespeicherten Master-Pages in einer Liste angezeigt werden. Über den Namen der zu bearbeitenden Master-Page kann diese ausgewählt und die entsprechenden Werte hierfür geändert werden. Nachdem die Werte geändert und gespeichert wurden, kann der Anwender über die Generierung des PDF-Dokuments die Auswirkungen auf den Master betrachten und gegebenenfalls wieder verändern.

Master-Pages entfernen

Der Anwender soll eine Master-Page aus dem System entfernen können. Die Master-Pages, die durch ihn für das Projekt angelegt wurden, sollten ihm dabei in einer Auswahlliste zur Verfügung stehen. Hier kann er nochmals sich die Werte der Master-Page über den Vorgang *Master-Page bearbeiten* betrachten, um sich somit zu vergewissern, dass auch die richtige Master-Page aus dem FO-Projekt entfernt werden soll. Die Master-Page soll in der Liste ausgewählt und entfernt werden. Hier gilt jedoch große Vorsicht. Wenn eine Master-Page aus dem System entfernt und diese für eine Seitensequenz verwendet wird, so kann es hier zu einigen Fehler kommen.

Um diese Fehler abzufangen, sollte der Anwender vor dem Löschen darauf hingewiesen werden, welche Auswirkungen dieser Prozess mit sich bringen würde und ob er sich sicher ist, dass er diese Master-Page aus dem Projekt entfernen möchte. Dabei wäre es möglich, die Namen der Seitensequenzen aufzuzeigen, die diesen Master verwenden. Somit würde er einen Überblick darüber bekommen, wo dieser Master verwendet wird, um gegebenenfalls die Seitensequenzen anzupassen.

5.4.2 Seitensequenzen bearbeiten

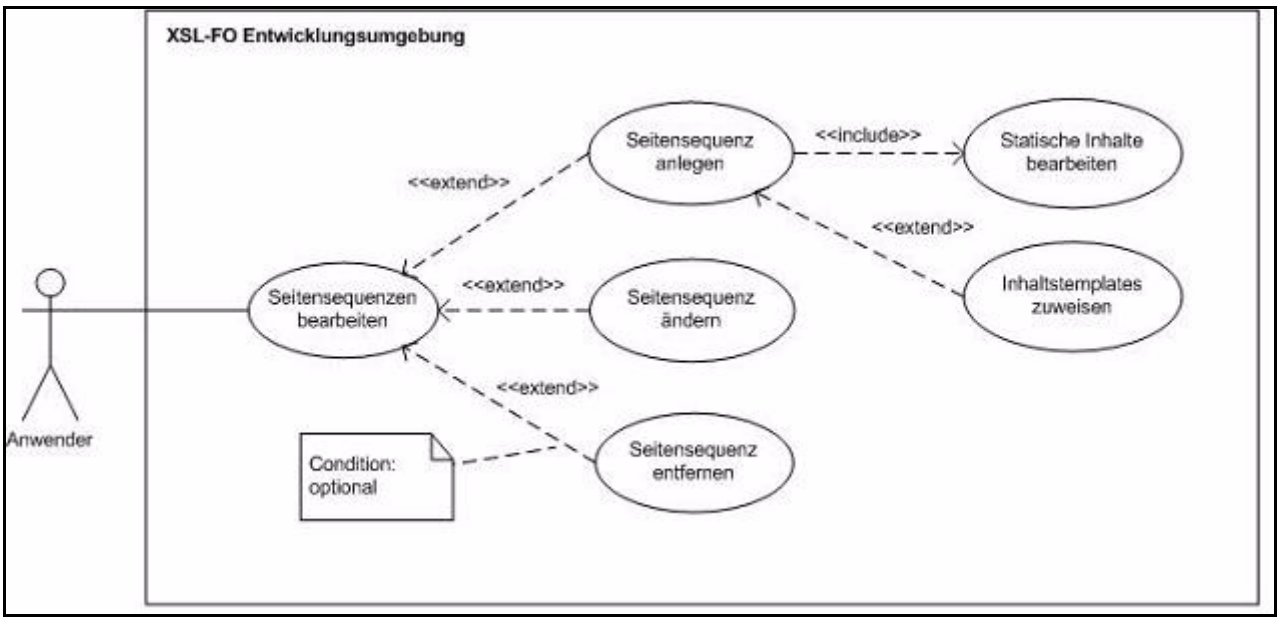


Fig. 5-12 : Use Case Diagramm, Seitensequenzen bearbeiten

Das Bearbeiten einer Seitensequenz beinhaltet die Use-Cases Seitensequenz anlegen, Seitensequenz ändern und Seitensequenz entfernen. Die Anforderungen hierzu werden in den folgenden Abschnitten näher beschrieben.

Seitensequenzen anlegen

Nachdem der Anwender nun seine Master-Pages definiert hat, sollte es ihm möglich sein, auf den eigentlichen Inhalt einer Seite näher einzugehen. An dieser Stelle, muss der Anwender noch kein Inhaltstemplate für die Seitensequenz vorweisen, welches die Elemente aus dem XML-Dokument für die Darstellung der Inhalte repräsentiert. Der Anwender soll durch die Anwendung dabei unterstützt werden, spezielle Templates in einem durch die Anwendung generiertes FO-Stylesheet dynamisch erzeugen zu lassen. Die Anwendung erzeugt ein XSL-FO Stylesheet, welches speziell auf den Anwender angepasst wird. Die Templates innerhalb dieses Stylesheets generieren eine Seitensequenz und referenzieren nun auf die zuvor im Master definierten Bereiche *Druckbereich*, *Kopfbereich*, *Fußbereich*, *linker* und *rechter Seitenbereich*.

Für diese Bereiche soll der Anwender nun statische Inhalte vergeben können. Es sollte möglich sein, Linien zu zeichnen, um eine Trennung der Bereiche zum Inhalt zu erzeugen. Ebenso könnten Grafiken, Seitennummerierungen, Kapitelüberschriften oder einfache Zeichenfolgen in diesen Bereichen platziert werden. Die kommende Abbildung verdeutlicht nochmals die verschiedenen Inhalte, die auf den meisten Seiten einer Dokumentation zu finden sind.

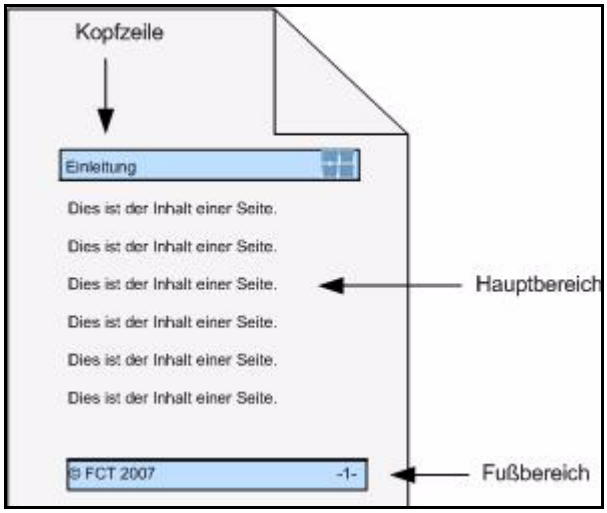


Fig. 5-13 : Seiteninhalte

In diesem Beispiel werden drei Bereiche hervorgehoben. Eine Kopf- sowie eine Fußzeile und der Hauptbereich. Dabei besteht die Kopfzeile aus zwei Elementen. Zum einen aus einer Grafik, die sich am rechten Rand des Bereichs befindet, sowie einer Kapitelüberschrift. Damit dieser Bereich vom Hauptbereich getrennt wurde, kam noch eine Linie hinzu. Die Fußzeile sieht etwas anders aus. Sie besteht aus einer einfachen Zeichenfolge, linksbündig ausgerichtet und einer Seitenzahl rechts außen. Die Fußzeile ist ebenso durch eine horizontale Linie vom Hauptbereich getrennt.

Diese Seite ist nur ein Muster, welches häufig in PDF-Dokumenten zu finden ist. Ebenso sollte die Möglichkeit gegeben werden, dass der Anwender die Anzahl der Elemente in den Bereichen selbst treffen kann. Es gibt durchaus auch Kombinationen, in denen nur einzelne Elemente oder gar mehrere Elemente in einer Kopf- bzw. Fußzeile auftreten können. Bei der Erstellung einer Seitenfolge sollte der Anwender auch hier einen eindeutigen Namen vergeben können, der für den Prozess innerhalb der FO-Entwicklung von Bedeutung ist.

Wenn der Anwender nun eine neue Seitensequenz in sein Projekt eingliedern möchte, so wäre es von Vorteil, wenn die Reihenfolge dieser Seiten verändert werden kann. Beim Anlegen einer Seitensequenz bezieht sich ja nun das Seitenlayout auf die zuvor angelegten Master-Pages. Somit sollten die Master-Pages zur Auswahl stehen, um den Seitensequenzen diese Seitenvorlagen zuweisen zu können, damit die PDF-Seiten des Dokuments das entsprechende Seitenlayout erhalten. Wenn die entsprechende Seitenvorlage zugewiesen wurde, so ist es nun möglich, jeden einzelnen Bereich des Masters mit einem Inhalt zu besetzen. Die Beschreibung dieser Anforderung wird im kommenden Absatz beschrieben.

Statische Inhalte bearbeiten

Das Bearbeiten von statischen Inhalten läuft wie folgt ab: Jede Master-Page Sequenz besitzt Referenzen auf die jeweiligen Kopf- und Fußzeilen, sowie auf linken und rechten Seitenbereich. Wenn der jeweilige Master ausgewählt wird, so sollten die Namen der zuvor konfigurierten Bereiche in einer Liste zur Verfügung gestellt werden, um diese dann bearbeiten zu können. Es gibt bspw. eine Kopfzeile mit dem Namen *header*. Dieser Bereich kann nun über gewisse Einstellungen definiert und mit Inhalten befüllt werden. Statische Inhalte bieten in XSL-FO die Möglichkeit, Abstände, Absatzformate und Ränder definieren zu können. Das bedeutet, dass dem Anwender diese Möglichkeiten über die Anwendung ebenfalls geboten werden muss, um hier auf diese Eigenschaften Einfluss zu nehmen. Er kann den Zeilenabstand nach unten bzw. oben regeln, er kann ein

gewisses Absatzformat auswählen, Rahmen um diesen Bereich setzen und bestimmte Elemente platzieren. Da, wie bereits erwähnt, der Anwender Absatzformte definieren kann, könnte er an dieser Stelle aus seinen bereits vorhandenen Absatzformaten eines auswählen. Nach der Generierung des PDF-Dokuments kann er betrachten, wie sich dieses Format auf den Bereich auswirkt und gegebenenfalls bestimmte Parameter an diesem Format einfach und bequem über die Anwendung ändern.

Die meisten Kopf- und Fußzeilen einer Dokumentation haben bestimmte Linien zur Hervorhebung des Bereichs. Solche Linien sollten natürlich ebenso vom Anwender definiert werden können. XSL-FO bietet hier die Möglichkeit, untere, obere, links- und rechtsseitige Ränder für diese Bereiche zu definieren. Somit könnte man einen Rahmen um den ganzen Bereich zeichnen lassen. Diese Möglichkeit kommt dem Benutzer sehr zu Gute, da er hier seinen Vorstellungen freien Lauf lassen kann und nicht nur für die Kopf- und Fußzeile jeweils unten oder oben eine Linie zeichnen kann. Um eine Linie zu zeichnen, sollte er die Linienart, Linienstärke und gegebenenfalls die Linienfarbe auswählen können. Nachdem nun diese Einstellungen vom Anwender abgearbeitet wurden, sollen nun Inhalte platziert werden. Wenn man eine Seite genauer betrachtet, dann fallen einem die vier Bestandteile Zeichenfolgen, Grafiken, Kapitelüberschriften und Seitenzahlen auf. Diese Möglichkeiten sollten über die Anwendung definierbar gemacht werden.

Für die Auswahl einer Grafik kann der Anwender den Pfad zu der Ressource angeben bzw. das Dateisystem durchsuchen lassen. Ebenso sollte er hier die Möglichkeit besitzen, die Ausrichtung der Grafik anzugeben, bspw. rechts- oder linksbündig sowie zentriert. Damit die Grafik in den einzelnen Bereichen angepasst weden kann, sollte die Höhe und die Breite dieser Grafik durch den Anwender frei einstellbar sein. Dabei könnten Maßeinheiten wie Prozent, Zentimeter oder Millimeter zur Auswahl stehen. Das brächte den Vorteil, dass der Anwender diese Angabe frei wählen kann und somit nur die Zahl ohne Maßeinheit anzugeben braucht.

Wenn die Auswahl auf eine einfache Zeichenfolge fällt, so sollte er diese über eine dementsprechende Eingabemaske eingeben können. Es sollte genügend Fläche vorhanden sein, damit man nicht immer wieder nach rechts oder nach unten über die Navigationspfeile navigieren muss. Somit bleibt die Eingabe immer im Blickfeld des Betrachters. Auch für diese Zeichenfolge sollte der Anwender die Ausrichtung angeben können. Zusätzlich zu links-, rechtsbündig und zentriert könnte es noch die Auswahl Blocksatz geben. Jedoch wäre dies auch denkbar bei den Einstellungen des Absatzformates, das dann für den Bereich zugewiesen werden kann.

Wenn der Anwender eine Seitenzahl einfügen möchte, so muss er hierzu nur die Position mit angeben können. Da die Darstellung einer Seitenzahl eindeutig ist, sind hiermit keine weiteren Einstellungen von Belang. Es gibt jedoch einige Varianten, wie man Seitenzahlen einfügen kann. Das folgende Beispiel zeigt einige davon.

```
a) Seite 1/10
b) page 5
c) - 2 -
d) 1/10
e) 1 von 10
```

Fig. 5-14 : Kombinationen von Seitenzahlen

Hier sieht man nun, wie viele Möglichkeiten es gibt, Seitenzahlen anzugeben. Im Verlaufe dieser Diplomarbeit sollte eine Möglichkeit gefunden werden, um dieses Spektrum auf das Wesentliche zu minimieren. Als letzte Möglichkeit können Ka-

pitelüberschriften als statische Inhalte beigefügt werden. Jede Kapitelüberschrift bezieht sich auf eine gesetzte Referenz im XSL-FO Stylesheet. Das bedeutet, dass der Anwender, um eine Kapitelüberschrift einsetzen zu können, wissen muss, auf welche gesetzte Referenz sich diese bezieht. Neben dem Namen dieser Referenz ist es wichtig, Position und Auftreten beim Eintreffen dessen sowie die Ausrichtung mit angeben zu können, damit diese auch richtig von allen drei FO-Prozessoren unterstützt werden kann. Das bedeutet, dass es hier ebenso Unterschiede in vielen Dokumentationen gibt. Eine Kapitelüberschrift kann sich über viele Seiten hinweg auf jeder Seite erstrecken, während es Beispiele gibt, da taucht diese Überschrift nur einmal beim ersten Vorkommen eines neuen Kapitels auf.

Inhaltstemplates zuweisen

Bis jetzt hat der Anwender nur statische Inhalte für die Seite einpflegen können. Nun soll er dabei unterstützt werden, komplette Informationen, die sich im Druckbereich der Seite befinden, einpflegen zu können. Hierbei soll nun auf die vom Power-User bereitgestellten Inhaltstemplates verwiesen werden. Das bedeutet, jedes dieser Templates stellt einen Inhalt dar. Um auf diese Inhaltstemplates zugreifen zu können, soll nun eine Lösung gefunden werden, die bequem über die Anwendung erfolgen kann. Man könnte sich vorstellen, dass der Anwender nun nachdem er über die Projekteinstellungen das XSL-FO Stylesheet mit angegeben hat, eine Auswahl zur Verfügung gestellt bekommt, über die er das jeweilige Inhaltstemplate auswählen kann. Wie dies jedoch dann umgesetzt werden soll, wird in der konzeptionellen Phase dieser Diplomarbeit ausgearbeitet.

Seitensequenz ändern

Nachdem eine Seitensequenz durch den Anwender angelegt wurde, sollte diese auch wieder verändert werden können. Hierbei sollte dem Anwender eine Liste mit vorhandenen Seitensequenzen zur Verfügung gestellt werden, über die er dann durch eine Auswahl eine beliebige Sequenz bearbeiten kann. Nachdem er nun diese ausgewählt hat, kann er die Werte verändern und über die PDF-Generierung das Ergebnisdokument betrachten. Sollten die Änderungen nicht in seinen Vorstellungen liegen, kann er diesen Schritt wiederholen. Durch das Ändern der jeweiligen Werte, welche vom System dann gespeichert werden, muss der Entwickler nun nicht mehr ständig das XSL-FO Stylesheet anpassen, um dort die statischen Inhalte von Hand zu verändern. Dies sollte nun alles über die Oberfläche der Anwendung möglich sein.

Seitensequenz entfernen

Sowie das Bearbeiten einer Seitensequenz gegeben sein sollte, so muss das Entfernen dieser ebenso über die Anwendung bereitgestellt werden können. Hierbei sollte der Anwender aus einer vorgegebener Liste vorhandener Seitensequenzen, die jeweilige aussuchen können, um diese aus dem akutellen Projekt zu entfernen. Daraufhin sollte der Anwender nochmals bestätigen, ob er diese Wahl auch tatsächlich treffen möchte. Sollte dies zutreffen, wird die komplette Seitensequenz aus dem generierten XSL-FO Stylesheet entfernt. Somit wird die Seitensequenz auch im Ergebnisdokument nicht mehr dargestellt, was bedeuten würde, dass die Seiten im PDF-Dokument und im FO-Dokument fehlen würden. Das Entfernen der Seitensequenz hat hier keine große Auswirkung wie das Entfernen einer Master-Page, da die Seitensequenz unabhängig laufen kann, und durch keine andere Einstellungsmöglichkeit verwendet wird. Somit kann das Entfernen dieser ohne Fehlermeldungen erfolgen.

5.4.3 Absatzformate bearbeiten

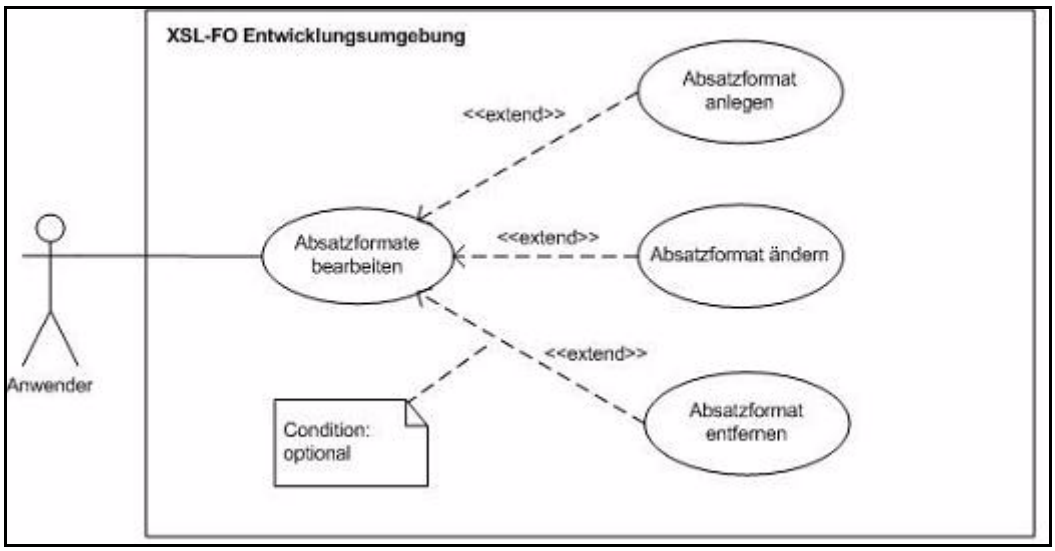


Fig. 5-15 : Use-Case Diagramm, Absatzformate bearbeiten

Das Bearbeiten eines Absatzformats beinhaltet die Use Cases Absatzformate anlegen, Absatzformate ändern und Absatzformate entfernen. Die Anforderungen hierzu werden in den folgenden Abschnitten näher beschrieben.

Absatzformate anlegen

Absatzformate besitzen eine vielseitige Anzahl an konfigurierbaren Parametern. Es soll hier auf die wichtigsten eingegangen werden. Nachdem der Anwender den Seitensequenzen seine definierten Master-Pages zugeordnet hat, hat er nun die Möglichkeit, Absatzformate zu definieren. Jedoch kann er auch ohne diesen Schritt in dem Verarbeitungsprozess fortschreiten. Um ein Absatzformat zu erstellen, muss ein passender Name hierfür verwendet werden. Er sollte so gewählt werden, dass eine einfache Identifikation darüber den Sinn dieses Formats wiederspiegelt. Die Definition eines Absatzformats bringt den Vorteil, dass sowohl für die einzelnen Bereiche einer Seite, als auch für die Inhaltstemplates im XSL-FO Stylesheet diese Formate verwendet werden können. Wenn der Anwender ein neues Absatzformat definiert hat, so sollte es möglich sein, dieses auch für den eigentlichen Seiteninhalt verwenden zu können. Somit muss sich der Power-User, der dann die Inhaltstemplates anpasst, zusammen mit dem Anwender über die Verwendung im Klaren sein. Der Power-User greift dann auf das entsprechende Absatzformat über dessen Namen zu und kann dann das angepasste Inhaltstemplate dem Anwender wieder zur Verfügung stellen.

Neben dem Namen sollte der Anwender die Auswahl einer Schriftart treffen können. Der Anwender könnte hier aus einer Liste vorgegebener Schriftarten die passende auswählen. Jedoch unterstützen nicht alle FO-Prozessoren dieselben Absatzformate. Wie aus der Analyse der FO-Prozessoren bereits herausgearbeitet wurde, kennt der Apache FOP das Format Arial nicht und benötigt hierzu eine weitere Konfigurationsdatei. Das bedeutet, dass eine Liste zur Auswahl dieser Formate nicht das Beste wäre, sondern der Anwender selbst den Namen hierfür vergeben sollte. Das heißt wiederum auch, dass der Anwender wissen sollte, wie die entsprechende Schriftart heißt, um diese dann auf das Absatzformat zu beziehen. Es gibt Schwierigkeiten bei der Benennung dieser Schriftarten. Es gibt zum einen Courier und Courier New. Der eine FO-Prozessor unterstützt Courier, der andere wiederum kennt nur Courier New. Dies sollte dann dem Anwender verdeutlicht werden. Damit keine Fehler bei den Schriftarten auftreten,

sollten diese aus dem System ermittelt werden, damit auch nur diese richtig angezeigt werden können. Das würde aber bedeuten, dass bei Generierung der PDF-Dokumente auf anderen Arbeitsplätzen die gesetzten Schriftarten nicht erkannt bzw. nicht auf dem System vorhanden sein könnten.

Desweiteren gehört neben der Schriftart auch die Schriftgröße mit angegeben. Hier sollte eine Liste verschiedener Schriftgrößen angeboten werden, wie man es bereits auch bei der Benutzung von Microsoft Word gewohnt ist. Der Schriftstil, bspw. fett oder kursiv, sollte ebenso über eine Auswahlliste für den Anwender zur Verfügung gestellt werden. Dabei ist es wichtig, dass diese Schriftstile kombiniert werde können. So wie man es als Benutzer von Microsoft Word kennt, sollte es auch hier möglich sein, da es immer wieder Wörter gibt, die fett und gleichzeitig kursiv sind. Es wäre auch zu überlegen, ob man die Eigenschaften Unterstrichen, Hoch- bzw. Tiefgestellt, Uppercase oder Lowercase mit in diese Auswahlliste aufnehmen sollte, um noch mehr Einstellungsmöglichkeiten in die Anwendung zu bekommen. Jedoch sollte dabei beachtet werden, dass dies prozessorunabhängig laufen sollte und gerade bei diesen Einstellungen Probleme bei der Umsetzung mancher FO-Prozessoren auftreten können.

Neben diesen Formateigenschaften gibt es noch die Wort- und Zeilenabstände. Diese sollten auch für den Anwender einstellbar sein. Die Ausrichtung kann ebenso vom Anwender getroffen werden. Hierbei soll unterschieden werden zwischen linksbündig, rechtsbündig, zentriert oder Blocksatz.

Absatzformate ändern

Nachdem der Anwender ein Absatzformat definiert hat, sollte er dieses auch bearbeiten können. Hierzu sollte ihm eine Liste mit allen vorhandenen Absatzformaten zur Verfügung gestellt werden, wobei er das zu bearbeitende Format daraus auswählen kann. Es sollte eine Eingabemaske in die Oberfläche gebracht werden, in der die Einstellungen des Absatzformats enthalten sein sollten. Der Name des Absatzformats sollte dabei für den Anwender nicht mehr bearbeitet werden können, da dieser Name im FO-Dokument als Referenz auf dieses Format angewendet sein könnte. Der Anwender kann die restlichen Werte nach Belieben ändern und über die PDF-Generierung die Auswirkungen betrachten.

Absatzformate entfernen

Falls ein erstelltes Absatzformat dem Anwender nicht zusprechen sollte, dann hat er die Möglichkeit, die Werte zu ändern oder es ganz aus dem System zu entfernen. Dazu sollte der Anwender aus einer Liste mit vorhandenen Absatzformaten das zu entfernende Format auswählen können. Das System sollte den Anwender daraufhin nochmals aufmerksam machen, ob das zu entfernende Format auch tatsächlich entfernt werden soll, da es sein kann, dass diese Format in bestimmten Bereichen einer Seitensequenz sowie im XSL-FO Stylesheet mit den Inhaltstemplates vorkommen kann. Das Entfernen des Formats kann dann zu einigen Problemen und Fehlermeldungen führen. Eine Liste mit den dementsprechenden Seitensequenzen bzw. Inhaltstemplates, die dieses Format verwenden, sollte dem Anwender dargestellt werden.

5.4.4 PDF generieren

Über die Anwendung sollte es möglich sein, anhand der gespeicherten Konfigurationen ein PDF-Dokument zu generieren. Damit dies auch funktioniert, muss der Anwender zumindest eine Master-Page, sowie eine Seitensequenz erstellt haben. Um ein PDF-Dokument generieren zu lassen, sollte der Anwender das XSL-FO Stylesheet mit den Inhaltstemplates, sowie sein XML-Dokument über die Projekteinstellungen angeben können. Der Speicherort des erzeugten FO-Dokuments sowie ein passender FO-Prozessor sollten ebenso über die Anwendung angegeben werden. Aus dem erzeugten FO-Dokument erzeugt dann der FO-Prozessor das PDF-Dokument. Dieses PDF-Dokument sollte dem Anwender zur Qualitätssicherung angezeigt werden.

5.5 FO-Projekt auf andere Server portieren

Um die Flexibilität der Anwendung zu fördern, sollte es dem Anwender möglich sein, die angelegten Master-Pages, Seitensequenzen und Absatzformate auf jedes beliebige System bzw. Server zu übertragen. Das bedeutet, dass erzeugte Batch-Dateien, die für die Erstellung des FO-Dokuments und des PDF-Dokuments zuständig sind auf anderen Servern lauffähig gehalten werden sollten, so dass diese dort nur noch ausgeführt werden, um die gewünschten Ergebnisdokumente zu erhalten. Die Batch-Datei für die Erzeugung des FO-Dokuments sollte dann nur noch das XML-Dokument mit den neuen Inhalten übergeben bekommen, damit dann auf der aktuellen Version das PDF-Dokument erzeugen kann. Über die XSL-FO Entwicklungsumgebung selbst werden zuvor die jeweiligen Konfigurationen definiert. Diese Einstellungen werden dann dementsprechend gespeichert. Die Batchprozesse können dann ohne die Anwendung laufen und das Ergebnisdokument erzeugen.

5.6 Standardtemplates bereitstellen

Bei der Auslieferung des Produkts an die Kunden von FCT werden XSL-FO Stylesheets mit Standardtemplates bereitgestellt. Diese Stylesheets verarbeiten XML-Dokumente mit der Kundenstruktur. Sollten Änderungen an der Struktur der XML-Dokumente auftreten, z.B. wenn neue Elemente oder Definitionen in die Struktur aufgenommen wurden, so müssten diese Templates dementsprechend angepasst werden, damit die neuen Elemente dementsprechend umgesetzt werden können. In der Auslieferung sind diese XSL-Stylesheets schon dabei und somit für den Kunden bereitgestellt. Jedoch befinden sich in diesen Stylesheets nur die Standardtemplates. Sollte es eine komplett neue DTD oder neue Elemente in der vorhandenen Struktur geben, so müssten diese Standardtemplates entsprechend angepasst werden.

5.7 Inhaltstemplates aus Standardtemplates ableiten

Sollten Änderungen an der Struktur des XML-Dokuments des Kunden vorgenommen werden, so müssen diese dementsprechend in dem XSL-FO Stylesheet vorgenommen werden. Der Power-User leitet dazu die Standardtemplates ab, um sie dann im weiteren Verlauf anzupassen.

5.8 Inhaltstemplates anpassen

Die abgeleiteten Standardtemplates werden nun dementsprechend auf die neuen Elemente angepasst. Hierzu muss der Power-User wissen, wie die entsprechend neuen Elemente umgesetzt werden sollen, bzw. wenn der Anwender bereits über die XSL-FO Entwicklungsumgebung neue Absatzformate definiert hat, diese dann in den Inhaltstemplates verwenden. Hierzu ist noch eine Absprache mit dem Anwender nötig, damit die Anforderungen umgesetzt werden können. Dazu benötigt der Power-User die Namen der jeweiligen neu hinzugekommenen Absatzformate um diese dann an den entsprechenden Stellen verwenden zu können.

5.9 Inhaltstemplates bereitstellen

Die angepassten Inhaltstemplates werden dann dem Anwender wieder bereitgestellt, um die Inhalte des XML-Dokuments auf die Seiten des PDF-Dokuments zu bekommen. Somit kann der Anwender dann auf die neuen Inhaltstemplates über die Entwicklungsumgebung zugreifen und diese den jeweiligen Seitensequenzen zuweisen.

6 Konzeption und Realisierung

Im Kapitel 4 wurden die Probleme, Unterschiede und Gemeinsamkeiten der FO-Prozessoren analysiert, um einen gemeinsamen Standard für alle drei Prozessoren schaffen zu können. Das Thema dieser Diplomarbeit ist eine prozessorunabhängige XSL-FO Entwicklungsumgebung, in der man ohne XSL-FO Kenntnisse das Seitenlayout eines PDF-Dokuments erstellen und über verschiedene FO-Prozessoren erzeugen kann. Die Schwierigkeit besteht darin, die Eingabeparameter so herauszufiltern, dass diese auch von den Prozessoren unterstützt werden bzw. zu untersuchen, welche Teile eines FO-Dokuments dynamisch erzeugbar sind und welche vom Anwender gepflegt werden müssen. Bei der Erzeugung eines PDF-Dokuments soll dieses eine Gemeinsamkeit aller drei Prozessoren aufweisen können, ohne dass sich das Layout einer Seite in vielen Punkten unterscheidet.

6.1 Konzeption der XSL-FO Entwicklungsumgebung

Es wird ein Konzept erarbeitet, aus dem heraus eine prozessorunabhängige XSL-FO Entwicklungsumgebung zur Unterstützung dieses Workflows entwickelt werden kann. In den folgenden Kapiteln soll darauf abgezielt werden, ob sich die in der Anforderungsanalyse beschriebenen Use Cases auch programmiertechnisch umsetzen lassen. Es soll herausgefunden werden, ob sich jedes dieser Kriterien umsetzten lässt. Wo Probleme bei der Umsetzungen auftreten, soll eine geeignete Lösung gefunden werden.

6.1.1 Ablauf zur Generierung des PDF-Dokuments

In der Anforderungsanalyse sind schon einige Komponenten angesprochen worden, die sich für den Ablauf zur Erzeugung eines gültigen FO-Dokuments besonders eignen würden. Um einen gesamten Überblick über diesen Verarbeitungsprozess zu bekommen, d.h. welche Komponenten in diesen Prozess inbegriffen sind, soll die Abbildung Fig. 6-1 Aufschluss geben. In der Analyse der FO-Prozessoren wurden zwei Verarbeitungsmöglichkeiten aufgezeigt, um aus einem FO-Dokument ein PDF-Dokument zu erzeugen. Die nachfolgende Abbildung zeigt die erste der beiden Möglichkeiten auf, in der über einen XSLT-Prozessor das FO-Dokument erzeugt wird, welches anschließend an einen entsprechenden FO-Prozessor überliefert wird, um das PDF-Dokument zu generieren. Bei diesem Prozess kann bei die Generierung das FO-Dokument bearbeitet werden, um eventuell bei Problemen des jeweiligen FO-Prozessors durch Debugging die entsprechende Stelle im Dokument zu finden und beheben zu können. Durch Batchprozesse kann dies über die Konsole nachvollzogen werden.

Die zweite Variante der PDF-Generierung liegt darin, den XSLT-Prozessor nicht einzusetzen, sondern den Vorgang direkt durch einen FO-Prozessor einzuleiten. Dies hat jedoch den Nachteil, dass das FO-Dokument indirekt erzeugt wird und der Anwender, je nach Option, nur das PDF-Dokument erhält. Die erste Variante hat den Vorteil, dass durch den XSLT-Prozessor das FO-Dokument erzeugt und im System des Anwenders gespeichert werden kann. Um das PDF-Dokument erzeugen zu können, gibt es nun einige Prozesse, die im weiteren Verlauf dieses Konzepts näher betrachtet werden. Um auf die Anforderungen an die XSL-FO Entwicklungsumgebung näher eingehen zu können, werden nun die einzelnen Systemkomponenten, die an dem Gesamtprozess beteiligt sind vorgestellt. Dabei soll geprüft werden, ob sich die gestellten Anforderungen programmiertechnisch umsetzen lassen.

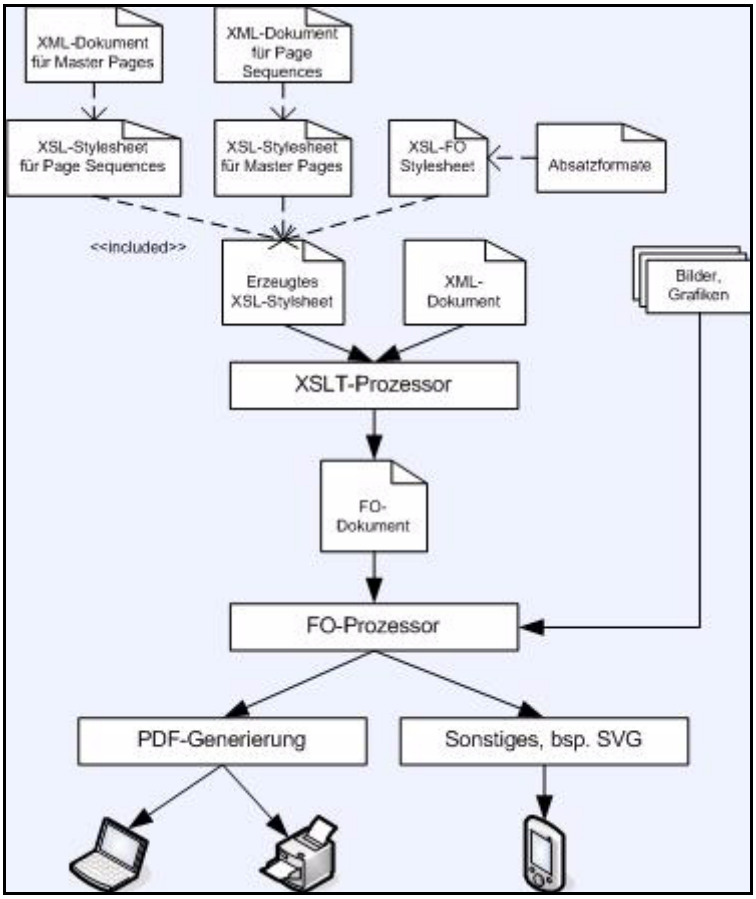


Fig. 6-1 : Verarbeitungsprozess vom FO- bis zum PDF-Dokument

6.1.2 Der Druckbereich zur Ausspielung der Seiteninhalte

Zur Ausspielung der Seiteninhalte wird ein XML-Dokument benötigt. Ohne das XML-Dokument können keine Inhalte beim Erzeugen des PDF-Dokuments auf die Seitenfolgen gebracht werden. Es gibt mehrere Möglichkeiten, um dieses XML-Dokument zu erstellen. Es muss jedoch immer eine logische Dokumentenstruktur enthalten. Zur Erstellung bietet sich ein Editor an, der XML unterstützt, bspw. Stylus Studio 2007[1]. Der Anwender sollte eine DTD einbinden, um den Aufbau der Dokumente einzuhalten. In der DTD sind alle Elemente definiert, die eingesetzt werden können, um ein gültiges XML-Dokument zu erzeugen. Problematiken wie Strukturverletzungen können vermieden werden.

Der Anwender erfasst nun den Inhalt der PDF-Seiten, die später über FO erzeugt werden, über einen entsprechenden XML-Editor. Sollte kein XML-Editor zur Verfügung stehen, so kann der Inhalt auch über einen einfachen Editor wie Notepad erfasst werden. Hier kann jedoch keine DTD eingesetzt werden, die erfasste Struktur ist somit nicht prüfbar. Der Strukturedeitor Adobe FrameMaker ist bereits bei vielen Kunden der Firma FCT im Einsatz. Diese erfassen damit ihre Technischen Dokumentationen und können über das TIM-RS Plugin einen XML-Export ansprechen. Das generierte XML-Dokument kann dann in der XSL-FO Entwicklungsumgebung bereitgestellt werden.

[1]http://www.stylusstudio.com

6.1.3 Anpassung und Bereitstellung der Inhaltstemplates

Ein wichtiges Dokument in diesem Verarbeitungsprozess ist das XSL-FO Stylesheet welches die Inhaltstemplates beinhaltet, die für die Darstellung der Seiteninhalte verwendet werden sollen. Das bedeutet, dass jedes Element aus dem XML-Dokument erfasst und verarbeitet werden muss. Es gibt gewisse Regeln in dem Stylesheet, welche die Inhalte aus den einzelnen Elementen herausfiltern und diese in ein FO-Element packen. Dazu muss der Entwickler dieses Dokuments sich mit der Sprache XSL-FO auseinandersetzten, damit er die einzelnen Bedeutungen der FO-Elemente versteht. Er muss wissen, welche Bedeutung die Elemente aus dem XML-Dokument besitzen, sowie ihre Darstellung mit XSL-FO. Hierzu ist eine Absprache mit dem Anwender notwendig, damit diese richtig umgesetzt werden.

Für das Root-Element des XML-Dokuments muss keine Regel aufgestellt werden, da diese im späteren Verlauf des Prozesses selbständig über die Anwendung ermittelt wird. In diesem Root-Element werden die Master-Pages und Seitensequenzen definiert, die durch Angaben des Anwenders über die XSL-FO Entwicklungsumgebung erzeugt werden. Somit wird dieses Inhaltstemplate nicht mehr in diesem Stylesheet benötigt. Bei den Auslieferungen an den Kunden werden die Stylesheets bereits mit ausgeliefert und zur Verfügung gestellt.

In diesen Stylesheets befinden sich Standardtemplates, die auf die Standardstruktur des XML-Dokuments des Kunden angepasst wurden. Das Seitenlayout und die Platzierung des Inhalts kann somit direkt vom Anwender erfolgen.

Das XSL-FO Stylesheet sollte von der Firma FCT entwickelt werden, da man dem technischen Redakteur kein Know-How in dem Bereich XSL-FO zumuten will. Der programmiertechnische Aufwand sollte beim Entwickler liegen und nicht beim Anwender. Der Anwender dagegen gestaltet über die XSL-FO Entwicklungsumgebung das Seitenlayout, Seitenformate und die Seitenfolgen Anschließend kann er auf jedes Inhaltstemplate in diesem Stylesheet bequem über die Oberfläche der Anwendung zugreifen, indem er das passende über eine Liste auswählt.

Sollte es Änderungen geben, so werden diese durch den Power-User angepasst und dem Anwender zur Verfügung gestellt. Änderungen könnten das Hinzufügen von neuen Elementen oder eine komplett neue DTD betreffen. Der Power-User leitet in diesem Fall die Inhaltstemplates aus den Standardtemplates ab und passt gegebenenfalls die Inhaltstemplates an. Bei der Verwendung unterschiedlicher FO-Prozessoren ist jedoch von Nachteil, dass für jeden dieser Prozessoren ein eigenes XSL-FO Stylesheet zur Ausspielung des Seiteninhalts entwickelt und angepasst werden muss. Andernfalls kann es hier zu Problemen kommen, da es Unterschiede bei der Unterstützung einiger Elemente gibt.

6.1.4 Konfigurationsdatei zur Bearbeitung der Master-Pages

Wenn verschiedene XSL-FO Projekte verwaltet werden sollen, so bietet sich für jede dieser Konfigurationen ein XML-Dokument an, welches Informationen zu den erstellten Master-Pages beinhaltet.

Bei der Bearbeitung der Master-Pages muss beachtet werden, welche Parameter umgesetzt werden können, bzw. welche Parameter nicht einstellbar sein sollten, da sie durch einige FO-Prozessoren nicht unterstützt werden. Die Analyse aus Kapitel 4 hat einen Überblick darüber gegeben, welche Unterschiede, Probleme und Gemeinsamkeiten bei der Verwendung der FO-Prozessoren bei Formatierungen eines FO-Dokuments auftreten können. Das Ziel dieser Diplomarbeit ist es, eine prozessorunabhängige XSL-FO Entwicklungsumgebung zu schaffen, das bedeutet, es soll mit allen FO-Prozessoren möglich sein, die selbe Ausgabe eines PDF-Dokuments hinzubekommen.

Damit jegliche Informationen zusammen kommen, muss das XML-Dokument dementsprechend aufgebaut sein. Da es mehrere Master-Pages geben kann, sollten diese auch alle hier aufgelistet sein. Die ersten konfigurierbaren Parameter sollten der Name, die seitlichen Abstände zu den Rändern sowie das Seitenformat sein. Dies sind zu aller erst einmal Eigenschaften, die von allen drei FO-Prozessoren unterstützt werden. Ein kleines Beispiel zeigt, wie man dies aufbauen könnte.

```
<MasterPagesParameter>
    <SimplePageMaster>
        <MasterName>...</MasterName>
        <Margin>10mm 7mm 7mm 18mm</Margin>
        <PageHeight>...</PageHeight>
        <PageWidth>...</PageWidth>
        ...
    </SimplePageMaster>
    .........
</MasterPagesParameter>
```

Fig. 6-2 : Erste Konfiguration einer Master-Page mit XML

Um ein gültiges XML-Dokument zu erhalten, ist es notwendig, dass immer nur ein Root-Element existiert. Hier wäre es das Element `MasterPagesParameter`. Das Element `SimplePageMaster` beinhaltet nun Informationen zu den einzelnen Master-Pages. Damit eine Master-Page über die Anwendung bearbeitet werden kann, wird anhand des Namens diese Vorlage herausgefiltert und die Daten an der Oberfläche dargestellt. Zur Bearbeitung einer Master-Page sollte der Anwender eine Auswahl mit den Namen der vorhandenen Master-Pages bekommen. Das bedeutet also, dass auch der Name der Master-Page im XML-Dokument stehen sollte, damit dieser für das Aussehen der jeweiligen Seiten referenziert werden kann. Im obigen Beispiel wird das Element `MasterName` verwendet.

Nun sollten die seitlichen Abstände der Ränder konfiguriert werden. Hierfür gibt es das Attribut `margin`, über das die seitlichen Grenzen einer Master-Page eingestellt werden können. Dazu bietet sich ein Element an, in dem die jeweiligen Werte eingetragen und später in dem Attribut stehen werden. Dies ist nur eine Möglichkeit. Über die Anwendung wird es vier Eingabefelder geben, für den Abstand nach oben, links, unten und rechts. Diese Werte werden dann zu einem Wert zusammengefasst und in das Element eingetragen. Anstelle des Elements `Margin` bietet sich noch folgende Möglichkeit an.

```
...
    <Margin_Top>10mm</Margin_Top>
    <Margin_Left>7mm</Margin_Left>
    <Margin_Bottom>18mm</Margin_Bottom>
    <Margin_Right>7mm</Margin_Right>
...
```

Fig. 6-3 : Margin-Elemente für Master-Pages mit XML

Man kann sich darüber streiten, welche Variante sich hier besser eignet. Die erste Variante würde sich eignen, wenn nur das Attribut `margin` im FO-Dokument zur Definition einer Master-Page verwendet werden würde. Wenn jedoch `margin-top`, `margin-left`, `margin-bottom` und `margin-right` stattdessen angewendet würden, so bietet sich die zweite Variante an.

Die erste Variante würde weniger Schreibarbeit und für mehr Übersichtlichkeit im FO-Dokument sorgen. Ein Problem, das sich daraus jedoch ergibt, ist, zu verstehen, welcher Wert für welchen seitlichen Abstand sorgt. Die Werte *top*, *left*, *bottom* und *right* sind schnell verständlich. Hier sollte doch die zweite Variante umgesetzt werden, da dies zu mehr Verständlichkeit führt, wenn der Code des FO-Dokuments einmal nachvollzogen werden sollte. Nachdem die Randeinstellungen konfiguriert wurden, sollte das Format einer Seite bestimmt werden können. Hierzu unterstützt jeder der FO-Prozessoren die Attribute `page-height` und `page-width`, die für die Höhe und Breite einer Seite zuständig sind. Hierbei bieten sich jeweils Elemente an, die diese repräsentieren. In Fig. 6-2 sind dies die Elemente `PageHeight` und `PageWidth`.

Mit diesen Einstellungen wäre das Grundgerüst einer Seite schon einmal komplett. Wenn man eine Master-Page erstellt hat, kann man diese zu einer Sequenz hinzufügen. Das bedeutet, dass bspw. ungerade und gerade Seiten zu einer Sequenz zusammengefasst werden. Das bietet sich für die eigentlichen Seiteninhalte an. Dabei kann man die Einstellungen treffen, über wieviele Seiten Master-Pages angewendet werden. Bei einem Inhaltsverzeichnis kann es beispielsweise sein, dass es nicht nur bei einer Seite bleibt. Solche Einstellungen sollten ebenso im XML-Dokument vorhanden sein. Diese könnten wie folgt aussehen.

```
<SimplePageMaster>
    ...
    <Type>single</Type>
    <SequenceReferenceName>...</SequenceReferenceName>
    <Options>
        <OddOrEven>...</OddOrEven>
        <PagePosition>...</PagePosition>
        <BlankOrNotBlank>...</BlankOrNotBlank>
    </Options>
    ...
</SimplePageMaster>
```

Fig. 6-4 : Zusatzeinstellungen für Master-Pages

Um zu wissen, welcher Sequenz diese Master-Page angehört, sollte hierzu ein Name vergeben werden, der dann im späteren Verlauf referenziert werden kann. Dies ist durch das Element `SequenceReferenceName` gegeben. Durch das Element `Type` kann der Typ festgelegt werden. In diesem Beispiel wird eine einfache Master-Page erzeugt, gekennzeichnet durch den Wert *single*. Zusätzliche Einstellungen, die bei einer Sequenz von Master-Pages vergeben werden können, sind, ob die Seite ungerade oder gerade ist, an welcher Position der Sequenz diese Master-Page auftreten soll bzw. ob diese Master-Page eine leere Seite

darstellt. Hier bietet sich ein Element `Options` an, welches Kindelemente bezüglich der genannten Einstellungen beinhaltet.

Desweiteren besteht eine Master-Page aus weiteren fünf Bereichen. Zum einen aus dem Hauptbereich, in dem die eigentlichen Inhalte dargestellt werden, des weiteren aus einer Kopf- und Fußzeile, sowie einem linken und rechten Seitenbereich. Für diese Bereiche sollte es separate Elemente geben, die die jeweiligen Einstellungen für diese repräsentieren. Die folgende Abbildung zeigt wie dies aufgebaut sein könnte.

```
<SimplePageMaster>
    ....
    <RegionBody>....</RegionBody>
    <RegionBefore>....</RegionBefore>
    <RegionAfter>....</RegionAfter>
    <RegionStart>....</RegionStart>
    <RegionEnd>....</RegionEnd>
</SimplePageMaster>
```

Fig. 6-5 : Fünf Bereiche einer Master-Page mit XML

Die fünf Elemente `RegionBody`, `RegionBefore`, `RegionAfter`, `RegionStart` und `RegionEnd` stellen hierbei jeweils diese Bereiche dar. Es ist darauf zu achten, dass die jeweiligen Elemente so benannt werden, dass sie leicht zu verstehen sind. Es bieten sich die Namen an, wie sie auch im FO-Dokument zu definieren sind. Nun sind eine ganze Reihe an Attributen für diese Bereiche definierbar. Zuerst werden Einstellungen bezüglich des Hauptbereichs untersucht. Hierbei kann der Hauptbereich wieder in den seitlichen Abständen eingeschränkt werden. Das bedeutet, dass angegeben werden kann, wie groß die Abstände von links, rechts, oben und unten sind. Hier gibt es wieder zwei Varianten, wie man die Grenzen einteilen kann. Anhand des unteren Beispiels erkennt man, dass dies durch das Element `Margin` erreicht werden kann.

```
<SimplePageMaster>
    ....
    <RegionBody>
        <Margin>....</Margin>
        <BackgroundColor>....</BackgroundColor>
    </RegionBody>
    ....
</SimplePageMaster>
```

Fig. 6-6 : Konfiguration des Hauptbereichs

Ein weiterer Faktor, der eine Rolle spielt, ist die Farbverwaltung. Die Auswahl der Hintergrundfarbe sollte dem Anwender zugänglich gemacht werden. Hierbei wählt er eine Farbe aus einer Liste aus. Dieser Wert wird bezogen und in einem Element gespeichert. Damit der Wiedererkennungswert erhalten bleibt, bietet sich der Name `BackgroundColor` an. In FO gibt es noch weitere Einstellungen für den Hauptbereich. Man könnte auch horizontale und vertikale Linien an den Rändern zeichnen. Jedoch werden diese Möglichkeiten nicht von allen FO-Prozessoren unterstützt und sollten somit in diesem Teil der Konfiguration noch nicht definiert werden.

Um jeden Bereich später referenzieren zu können, bedarf es eines geeigneten Namens. Die vier Bereiche können nun nicht wie der Hauptbereich von allen vier Seiten kommend eingegrenzt werden. Die Kopfzeile kann jeweils nach unten, die Fußzeile nach oben, der linke Seitenbereich nach rechts und der rechte Seitenbereich nach links eingeschränkt werden. Dabei gibt es das Attribut `extent`, welches bei diesen Elementen gesetzt werden kann. In der XML-Instanz würde sich daraus das Element `Extent` anbieten. Die untere Abbildung verdeutlicht die Konfiguration als Beispiel für die Kopfzeile. Alle anderen Bereiche besitzen dieselben Einstellungsmöglichkeiten.

```
<SimplePageMaster>
    ....
    <RegionBefore>
        <FlowName>header</FlowName>
        <Extent>...</Extent>
    </RegionBefore>
    ....
</SimplePageMaster>
```

Fig. 6-7 : Konfiguration einer Kopfzeile

Weitere Einstellungen werden wiederum nicht von jedem FO-Prozessor unterstützt.

6.1.5 Konfigurationsdatei zur Bearbeitung der Seitensequenzen

Ebenso wie für die Master-Pages, eignet sich ein XML-Dokument zum Konfigurieren der Seitensequenzen. Hierbei soll der Anwender dabei unterstützt werden, statische Inhalte in den vier Bereiche Kopfzeile, Fußzeile, linker und rechter Seitenbereich zu platzieren. Ebenso soll er angeben können, welches Template aus seinem XSL-FO Stylesheet für den Hauptbereich angezogen werden soll. Der Anwender kann einen geeigneten Namen für die Seitensequenz vergeben. Dieser Name wird über das Element `call-template` in einem generierten XSL-FO Stylesheet verwendet. In dem Stylesheet werden Templates angelegt, die für die Erzeugung der Seitensequenzen sorgen. Diese beziehen ihre benötigten Parameter aus dem Konfigurationsdokument und legen das Element fo:page-sequence im FO-Dokument an. Das folgende Beispiel zeigt, welche Parameter hierbei verwendet werden sollten.

```
<PageSequencesParameter>
    <Section>
        <NameOfTemplate>....</NameOfTemplate>
        <ReferenceToMasterPage>....</ReferenceToMasterPage>
        <ApplyTemplate>....</ApplyTemplate>
        <UseMode>....</UseMode>
        ...
    </Section>
</PageSequencesParameter>
```

Fig. 6-8 : Erzeugung von Templates über Konfig in Page Sequences

Neben dem Namen des zu erzeugenden Templates wird die Referenz auf die jeweilige Master-Page benötigt, um das Aussehen der Master-Page auf diese Seite abzubilden. Hier bietet sich ein Element an, welches den Namen des Elements aus dem XML-Dokument für die Master-Pages enthält. Somit hätten wir das Gerüst eines Templates. Damit der Anwender nun dem Hauptbereich Inhalt verleihen kann, sollte er angeben können, auf welches Template sich der Inhalt

bezieht. Dieses Template ist in seinem XSL-FO Stylesheet enthalten. Dabei gibt es zwei Varianten.

Zum einen kann ein Template über `xsl:apply-templates select=""` angesprochen werden. Zum anderen gibt es noch die Möglichkeit, einen Modus dabei anzugeben, dies würde dann der Zeile `xsl:apply-templates select="" mode=""` entsprechen. Somit ist der Anwender flexibel, was seinen Programmierstil angeht. Es sollte erreicht werden, dass der Anwender nicht in der Wahl dieser Einstellungen eingeschränkt bleibt. Für diese Einstellungen ist es notwendig eindeutige Elemente zu finden, die diese durch ihren Namen beschreiben. Das obige Beispiel ist eine Möglichkeit, um dieses so umzusetzen. Das Element `ApplyTemplate` gibt an, über welche Templates der Inhalt der Seite dargestellt werden soll. Das Element `UseMode` wäre dann für einen optionalen Modus des Templates.

Je nach Definition der Master-Pages können nun mehrere statische Inhalte für das Aussehen einer Seite erzeugt werden. Dazu bietet sich in der XML-Instanz ein Element an, welches diese Inhalte besitzt. In der unteren Abbildung gibt es das Element `StaticContents`, welches mehrere `Content` Elemente beinhalten kann. Jeder Bereich kann nun nochmals weiter eingeschränkt werden, was die seitlichen Abstände betrifft. Hierzu gibt es Abstände über und unter den Inhalten dieses Bereichs, sowie die Höhe. Für diese Angaben bieten sich wiederum einfache Elemente an, die diese Werte beinhalten.

```
<StaticContents>
    <Content>
        <PaddingBefore>....</PaddingBefore>
        <PaddingAfter>....</PaddingAfter>
        <ContentHeight>....</ContentHeight>
        <FlowName>footer</FlowName>
        ...
    </Content>
    ...
</StaticContents>
```

Fig. 6-9 : Abstände innerhalb eines Seitenbereichs

Durch den Namen der Elemente wird sofort klar, auf was sie sich im generierten FO-Dokument beziehen. Somit bleibt es gut leserlich und übersichtlich. Damit der Inhalt eines Bereichs deklariert werden kann, muss eine entsprechende Referenz auf den jeweiligen Bereich (Hauptbereich, Kopfbereich, Fußbereich, linker und rechter Seitenbereich) der Master-Page gesetzt werden können. Im oberen Beispiel wurde der Name *footer* eingegeben. Dieser Name bezieht sich auf den Bereich `region-after` einer Master-Page. Bei der Konfiguration der Master-Page wurde ebenso der Name *footer* mit angegeben, um somit eine Verbindung zum Design dieses Bereichs zu schaffen. Somit wären die äußeren Einstellungen solch eines Bereichs abgeschlossen.

Eine weitere notwendige Einstellung betrifft die Schrift- und Absatzgestaltung. Diese Einstellungen sind dem Anwender über die Oberfläche zugänglich zu machen und sollten dann ebenso im XML-Dokument für die Erzeugung der Seitensequenzen gespeichert werden können. Hierbei können Einstellungen der Schriftart, Schriftgröße, Schriftstil, Zeilenabstände und Schriftfarbe getroffen werden. Diese Angaben wären jedoch nicht zwingend im XML-Dokument notwendig, da sie bereits im XSL-Stylesheet für die Konfiguration der Absatzformate gespeichert werden. Jedoch ist der Name des Absatzformats hier wichtig, da über diesen Namen dann das Objekt über das Stylesheet ermittelt und instanziert werden kann. Ein Beispiel hierfür könnten folgendermaßen aussehen.

```
<ParagraphFormat>
    <Name>Cover_Title</Name>
</ParagraphFormat>
```

Fig. 6-10 : Schriftart und Absatzgestaltung bei statischen Inhalten

Hier gibt es das Element `ParagraphFormat`, welches das Element `Name` beinhaltet. Hier ist der Name des Absatzformates für den statischen Inhalt gesetzt worden. Der Name wird über die Eingabe des Anwenders in der Anwendung bezogen. Hierbei kann der Anwender aus einer Liste bereits im Projekt vorhandener Absatzformate das passende auswählen.

Bei der Bearbeitung einer Master-Page ist es notwendig gewesen, die horizontalen und vertikalen Linien nicht zu zeichnen, da sie in dieser Definition nicht von jedem FO-Prozessor unterstützt worden wären. Bei der Bearbeitung der statischen Inhalte ist es nun möglich, diese zeichen zu lassen. Ein Bereich besitzt obere, untere, linke und rechte Kanten. Jeder Kante kann ein Linienstil vergeben werden. Diese Einstellungen werden dem Anwender zugänglich gemacht und er kann somit einen Bereich völlig mit Linien umschließen. Auf die XML-Struktur bezogen ist es am besten, wenn es ein Element geben würde, welches aus den Kindelementen der jeweiligen Kante besteht, um so jede Kante individuell zeichnen zu können. Diese Struktur kann wie folgt aussehen.

```
<Borders>
    <TopSide>
        <Style>....</Style>
        <Width>....</Width>
        <Color>....</Color>
    </TopSide>
</Borders>
```

Fig. 6-11 : Liniengestaltung in den Seitensequenzen

Es sollte unterschiedliche Elemente zur Darstellung der Linien geben. Hier wird zum Zeichnen einer Linie im oberen Teil das Element `TopSide` verwendet. Es wird demnach die Elemente `BottomSide` für die Darstellung einer Linie im unteren Teil, `LeftSide` zur Darstellung einer Linie im linken Teil und `RightSide` zur Darstellung einer Linie im rechten Teil geben. Somit können diese eindeutig voneinander unterschieden werden. Eine Linie besitzt Eigenschaften bezüglich ihrer Linienfarbe, Linienstil und Linienstärke. Diese sollten ebenso als Kindelemente deklariert, bzw. abgespeichert werden.

Die Schwierigkeit, die sich nun ergibt, ist die Darstellung von Text und Bild in den Bereichen. Dabei richtet sich das Augenmerk besonders auf den Kopf- und Fußbereich einer Seite. Viele Kunden der Firma Fischer Computertechnik haben in ihrer Kopfzeile mehrere Zeilen, die die jeweiligen Kapitelüberschriften betreffen. Dazu kommt noch ein Firmenlogo entweder direkt oben drüber oder auf der rechten Seite des Bereichs. Nun gilt es, hierzu eine Lösung zu finden, wie der Anwender diese Inhalte bequem und einfach eingeben kann. Damit der Anwender nicht in der Auswahl der Elemente beschränkt bleibt, sollte es ihm möglich sein, mehrere Elemente in den Bereichen eines statischen Inhalts zu platzieren. Elemente, die in diesen Bereichen platziert werden können sind Zeichenfolgen, Grafiken, Kapitelüberschriften (sogenannte Marker) und Seitenzahlen. Durch eine Analyse Technischer Dokumentationen von Kunden der Firma FCT, kamen diese Erkenntnisse zum Vorschein. Dies sind Elemente, die meistens in Kopf- und Fußzeilen auftreten. Die Anordnung der Elemente verläuft jeweils horizontal. Das bedeutet, die Elemente im PDF-Dokument werden von links nach rechts ausgegeben. Ein Beispiel soll diese Anordnungen zum besseren Verständnis aufzeigen.

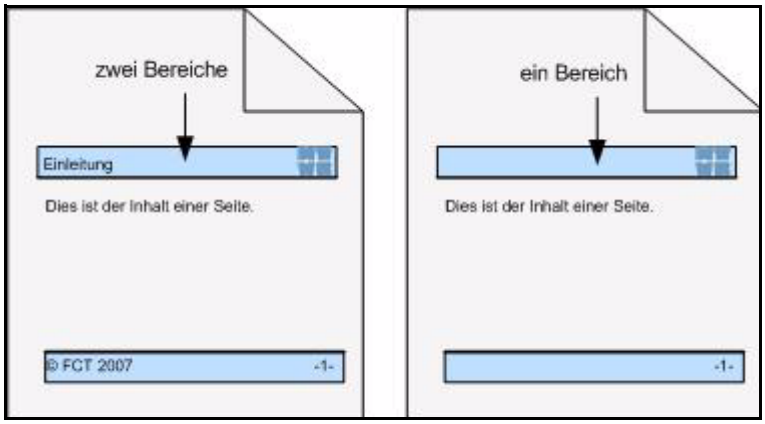


Fig. 6-12 : Horizontale Anordnung von Elementen in Kopf- und Fußzeile

In diesem Beispiel wurde eine Grafik rechts oben und ein Text links oben platziert. Desweiteren können auch Kapitelüberschriften auf der linken und eine Grafik auf der rechten Seite platziert werden. Es gibt einige Kombinationen, die nicht auf dem Bild zu sehen sind. Dies esind dann dem Anwender überlassen. Innerhalb dieser Trennung sollte es möglich sein, die Elemente nochmals zu positionieren, das bedeutet links- oder rechtsbündig, sowie zentriert oder im Blocksatz. Somit kann der Anwender die Elemente nach seinen Belieben an die passende Stelle setzen. Im XML-Dokument zur Konfiguration der Seitensequenzen könnte dies dann wie folgt aussehen.

```
<ContentElements parts="2">
    <ContentValue>....</ContentValue>
    <PageNumber>....</PageNumber>
</ContentElements>
```

Fig. 6-13 : Seiteneinteilung eines Bereichs

Damit die Anwendung später genau weiß, wieviele Elemente durch den Anwender festgehalten worden sind, wird das Attribut `parts` hierfür verwendet.

Nun wird es Elemente geben, in denen die Informationen stehen, die dann auf den entsprechenden Seitenfolgen ausgegeben werden sollten.

Wenn der Anwender eine Zeichenfolge über die Oberfläche eingeben würde, so bietet sich ein dementsprechendes Textfeld an. Der Kontext wird erfasst und in einem Element hinterlegt. Jetzt sollte der Anwender, wie bereits erwähnt, die Positionierung dieses Textes angeben können. Neben der Positionierung des Textes sind noch die Angaben zur Bestimmung der jeweiligen Größe und der Hintergrundfarbe des Bereichs von Bedeutung. Das entsprechende Element könnte dann folgendermaßen aussehen.

```
<ContentValue>
    <Value>FCT 2007 - erstellt mit TDM-RS</Value>
    <AlignHorizontal>left</AlignHorizontal>
    <AlignVertical>center</AlignVertical>
    <ElementSize>10cm</ElementSize>
    <BackgroundColor>rgb(255,255,255)</BackgroundColor>
</ContentValue>
```

Fig. 6-14 : Wortfolge-Element bei Seitensequenzen

Ein ebenso einfaches Element ist die Seitenzahl. Hierbei kann angegeben werden, wo sich die Seitenzahl befindet bzw. welche Größe und Hintergrundfarbe des Bereichs bezogen auf die Seitenzahl zugeordnet werden soll.

```
<PageNumber>
    <AlignHorizontal>right</AlignHorizontal>
    <AlignVertical>center</AlignVertical>
    <ElementSize>5cm</ElementSize>
    <BackgroundColor>rgb(255,255,255)</BackgroundColor>
</PageNumber>
```

Fig. 6-15 : Seitenzahl-Element bei Seitensequenzen

Hier wird das Element `PageNumber` definiert, welches das Element `AlignHorizontal` und `AlignVertical` zur Ausrichtung der Seitenzahl im jeweiligen Bereich angibt. Für die Größe des Bereichs wurde das Element `ElementSize` und für die Hintergrundfarbe das Element `BackgroundColor` eingesetzt. In diesem Beispiel wäre es jedoch nicht möglich, einen Text hinter oder gar davor mit anzugeben, bspw. für die Darstellung *Seite - 1 -* . Hierzu wäre es denkbar, zwei weitere Elemente mit aufzunehmen, die jeweils den Text davor und danach beinhalten würden. Es gibt noch den Fall, dass auch alle Seitenzahlen mit angegeben werden könnten, bspw. *Seite 1 / 10* . Für diese Einstellung könnte es ein zusätzliches Element geben, bspw. `UntilLastPage` welches die Werte */* oder *von* beinhalten könnte.

Um eine Grafik auszuwählen, muss der Pfad zu dieser Grafik angegeben werden können. Um nicht die Grafik eins zu eins in ihrer Größe übernehmen zu müssen, gibt man die Höhe und die Breite extra mit an. Natürlich kommt hier ebenso die Ausrichtung sowie Größe und Hintergrundfarbe des Bereichs der Grafik hinzu. Wenn der Anwender diese Einstellungen definiert hat, sollten hierfür folgende Elemente in der XML-Instanz zu finden sein (nur ein Beispiel).

```
<Graphic>
    <Src>.... source ....</Src>
    <Width>....</Width>
    <Height>....</Height>
    <AlignHorizontal>....</AlignHorizontal>
    <AlignVertical>....</AlignVertical>
    <ElementSize>100%</ElementSize>
    <BackgroundColor>rgb(255,255,255)</BackgroundColor>
</Graphic>
```

Fig. 6-16 : Grafik-Element bei Seitensequenzen

Das letzte Element das durch den Anwender konfiguriert werden kann, ist das sogenannte `Retrieve-Marker`-Element. In den Inhaltstemplates können bestimmte Marker gesetzt werden. Marker eignen sich für die Ausgabe von Kapitelüberschriften. Über den Namen des Markers kann der Anwender nun, meistens im Kopfbereich, Kapitelüberschriften erzeugen und angeben, ob diese bei jedem Beginn des Kapitels auf der jeweiligen Seite angegeben wird, bzw. an welcher Position dieser zu finden ist. Wie dies in der Konfigurationsdatei aussehen kann, zeigt die folgende Abbildung.

```
<RetrieveMarker>
    <RetrieveClass>....</RetrieveClass>
    <RetrievePosition>....</RetrievePosition>
    <RetrieveBoundary>....</RetrieveBoundary>
    <AlignHorizontal>right</AlignHorizontal>
    <AlignVertical>center</AlignVertical>
    <ElementSize>1cm</ElementSize>
    <BackgroundColor>....</BackgroundColor>
</RetrieveMarker>
```

Fig. 6-17 : Marker-Element bei Seitensequenzen

Die Benennung der Elemente ist mit den Attributs-Namen des jeweiligen `fo:retrieve-marker` Elements gekoppelt worden. Somit kann genau dieses Element dem jeweiligen Attribut zugeordnet werden.

All die genannten Konfigurationen für eine Seitensequenz können durch jeden FO-Prozessor unterstützt und umgesetzt werden. Es gibt noch zahlreiche weitere Attribute, die jedoch nicht von jedem FO-Prozessor gleich interpretiert werden. Die Anwendung wäre somit nicht prozessorunabhängig.

6.1.6 Konfigurationsdatei zur Bearbeitung der Absatzformate

Für das XSL-FO Stylesheet ist es wichtig, dass gewisse Absatzformatierungen flexibel gestaltet werden können. Über die Anwendung sollte der Anwender die Möglichkeit haben, Absatzformate zu erstellen, die in seinem Stylesheet auf gewisse Blöcke des FO-Dokuments referenziert werden können. Hierbei bietet sich wiederum XSL an, um die Absatzformate zu beschreiben. Über das Attribut `xsl:use-attriubte-sets` können definierte Attribute über ihren Namen bezogen werden. Wie dies zu verstehen ist, zeigt das kommende Beispiel.

```
<fo:block xsl:use-attribute-sets="H1_Heading">
    .....
</fo:block>
```

Fig. 6-18 : Beispiel mit xsl:use-attribute-sets

In diesem Beispiel wird ein Block erzeugt, welcher das Absatzformat *H1_Heading* verwendet. Der Inhalt dieses Blocks wird dann in dem jeweiligen gesetzten Format ausgegeben. Damit der Anwender nicht ständig diese Anpassung selbst in seinem XSL-FO Stylesheet vornehmen muss, wird dies über die Oberfläche flexibel regulierbar. Dieses Attribut ersetzt eine große Anzahl an Attributen, die dann in dem Element `fo:block` mit definiert werden müssten. Das wiederum ist mit weniger Schreibarbeit für den Anwender verbunden. Die Parameter müssten über die Anwendung gesetzt werden. Das generierte XSL-Dokument für die Absatzformate hat dann den folgenden Aufbau.

```
<xsl:attribute-set name="H1_Heading">
    <xsl:attribute name="font-family">Arial</xsl:attribute>
    <xsl:attribute name="font-size">16pt</xsl:attribute>
    <xsl:attribute name="font-weight">bold</xsl:attribute>
    <xsl:attribute name="font-style" />
    <xsl:attribute name="text-decoration" />
    <xsl:attribute name="line-height">1pt</xsl:attribute>
    <xsl:attribute name="space-before">1pt</xsl:attribute>
    <xsl:attribute name="space-after">1pt</xsl:attribute>
</xsl:attribute-set>
```

Fig. 6-19 : Absatzformate mit xsl:attribute-set

Hier sind Einstellungen bezüglich Schriftart, Schriftgröße, Schriftstil, Zeilenabstände sowie Wortabstände getroffen worden. Diese Einstellungen würden jetzt für den obigen Block gelten. Ohne diese Flexibilität würde der Entwickler den Block wie folgt beschreiben müssen.

```
<fo:block font-family="Arial" font-size="16pt"
          font-weight="bold" line-height="1pt"
          space-before="1pt" space-after="1pt">
    .....
</fo:block>
```

Fig. 6-20 : fo:block mit Attributen für Absatzgestaltung

Hier sieht man, wieviele Attribute der Entwickler an diesem Element selbst setzen muss, damit er die gewünschte Formatierung für den Block bekommt. Natürlich kann der Entwickler nur auf diese Attribute zugreifen, wenn diese existieren. Andernfalls würde es eine Warnung geben. Damit die Attribute referenziert werden können, muss der Entwickler wissen, wo sich dieses Stylesheet befinden, um es in seinem Dokument über das Element `xsl:include` bekannt zu machen. Dieser Pfad sollte dann in einer Konfigurationsdatei stehen, um diesen nicht immer fest setzen zu müssen. Die Benutzung dieser Absatzformate kann lediglich in den Bereichen der statischen Inhalte genutzt werden.

Sollten diese Absatzformate im Druckbereich der Seite eingesetzt werden, so müsste der Power-User das XSL-FO Stylesheet anpassen und wissen, wie die jeweiligen Formate in diesem Stylesheet benannt wurden, um diese dann verwenden zu können. Der technische Redakteur würde somit dem Power-User mitteilen, welche neuen Absatzformate er mit der XSL-FO Entwicklungsumgebung angelegt hat, um diese mit in die Ausgabe des Druckbereichs zu bekommen. Der Power-User kann über das Attribut `xsl:use-attribute-sets` auf den Namen des neu angelegten Absatzformats zugreifen und dieses nutzen. Das Inhaltstemplate wird somit angepasst und dem Anwender wieder zur Verfügung gestellt.

6.1.7 Verarbeitungsprozess der Master-Pages

Da die Master-Pages in XML gespeichert wurden, bietet sich für die Verarbeitung ein XSLT-Stylesheet an, das aus den zugrundeliegenden Master-Pages jeweils FO-Elemente generiert. Hierzu muss das Stylesheet jeden Eintrag durchgehen und anhand der Elemente erkennen, welches für den Hauptbereich, Kopfbereich, Fußbereich, linken und rechten Seitenbereich gesetzt wurde. Diese Bereiche müssen durchlaufen werden, um alle Werte zu bekommen und umzusetzen. Je nach gesetztem Wert wird der Bereich angelegt.

Alle Informationen, die im XML-Dokument stehen, müssen über dieses Stylesheet verarbeitet werden. Im Stylesheet gibt es ein Template, dass in jedes andere Stylesheet inkludiert werden kann, um diese Funktion aufzurufen. Die Funktion liefert dann das komplette `fo:layout-master-set` Element zurück, welches die Master-Pages enthält. Diese selbständige Generierung der Master-Pages hat den Vorteil, dass die Anwendung nicht selbst die Elemente erzeugen muss. Somit wird dies alles von den jeweiligen Stylesheets geregelt und dynamisch erzeugt. Die Anwendung selbst müsste nur die XML-Dokumente erzeugen, um so die Grundlage für die Stylesheets zu liefern, die diese XML-Dokumente verarbeiten können. Das Stylesheet sollte anhand der Informationen aus dem XML-Dokument ebenso die Sequenzen von mehreren Master-Pages zusammenfassen und erzeugen können. Hierbei weiß jede Master-Page, zu welcher Sequenz sie gehört und ob diese Master-Page für einzelne oder mehrere Seiten gilt.

XSL-FO bietet die Möglichkeit Kombinationen von einzelnen und sich wiederholenden Seiten darzustellen. Diese Möglichkeiten sollten dem Anwender über die Oberfläche angeboten werden, die dann über das Stylesheet aus dem XML-Dokument verarbeitet werden.

6.1.8 Verarbeitungsprozess der Seitensequenzen

Da die Konfigurationsdaten der Seitensequenzen in einer XML-Datei gespeichert wurden, bietet sich für deren Verarbeitung ebenso ein XSLT-Stylesheet an. Dieses Stylesheet sollte sich um die statischen Inhalte einer Seitenfolge kümmern. Das heißt, es muss die Inhalte aus dem jeweiligen XML-Dokument herausfiltern, um diese in FO darzustellen. Dieses Stylesheet läuft nicht selbständig ab, sondern agiert zusammen mit dem generierten Stylesheet, welches im darauffolgenden Kapitel näher betrachtet wird. Die Aufteilung der vier Bereiche eines Druckbereichs einer Seite, hat nun den Vorteil, dass Tabellen in FO erzeugt werden können. Je nach Angabe des Anwenders hat diese Tabelle eine oder mehrere Spalten, die jeweils mit den Elementen gefüllt werden.

Der wesentliche Vorteil, der sich hier durch die Gestaltung von Tabellen anbietet, ist, dass Linien für die jeweiligen Bereiche gezeichnent werden können. Jede Tabelle hat die Möglichkeit, Ränder zu zeichnen. Die Definition einer Linie hatte im Bereich der Master-Page keinen Sinn, da hier die Linien sowie die Grafiken nicht von allen FO-Prozessoren unterstützt wurden. In diesem Bereich gibt es nun die Möglichkeit, über die Tabellen, die Ränder als Trennlinien zu benutzen. Auch Grafiken können hier von allen FO-Prozessoren unterstützt werden. Zu erst einmal muss geprüft werden, wieviele Spalten der statische Inhalt enthalten soll. Dies wird anhand eines Elements festgelegt. Somit kann über eine For-Schleife das Element `fo:table-column` je nach Angabe der Spaltenanzahl erstellt werden.

Wenn der Bereich nur aus einem Teil besteht, dann muss in der Zeile auch nur eine Spalte definiert und erzeugt werden, für zwei Spalten jeweils zwei. Je nach Anzahl muss in jeder Spalte nun herausbekommen werden um was für ein Element es sich handelt. Hierbei helfen Templates, die jeweils den gesamten Kno-

ten übergeben bekommen. In diesem Template wird dann überprüft, ob es sich bei diesem Element um eine Grafik, eine Zeichenfolge, eine Seitenzahl oder einen Marker handelt. Je nach Element, wird dies entsprechend in FO dargestellt. Somit würde man die statischen Inhalte erzeugt bekommen.

6.1.9 Generierung des XSL-FO Stylesheets

Dieses Stylesheet wird zusammen mit dem XML-Dokument des Anwenders an einen XSLT-Prozessor geschickt, der daraus dann das FO-Dokument erzeugt. Bei jeder Generierung des PDF-Dokuments wird das Stylesheet neu generiert. Damit das Stylesheet einwandfrei durchlaufen werden kann, sollte es wissen, wo die beiden XSL-FO Stylesheets zur Erzeugung der Master-Pages und zur Erzeugung der Seitensequenzen im System liegen. Diese Pfade werden aus der Konfigurationsdatei ermittelt.

Ebenso sollte es den Pfad samt Dateinamen des XSL-FO Stylesheets, welches die Inhaltstemplates besitzt, kennen. Dies wird automatisch bekannt gegeben, da der Anwender dieses bereits zur Generierung seiner Vorschau der Anwendung mitteilen muss. Somit hat das Stylesheet jegliche Referenz auf die benötigten Dokumente, um das FO-Dokument ohne Probleme erzeugen zu können.

In diesem Stylesheet gibt es ein einziges Template mit dem Match auf das Root-Element des XML-Dokuments (strukturierte Inhalte). Hier gibt es mehrere Varianten, dies zu ermitteln. Zum einen kann das Root-Element aus dem angegebenen XML-Dokument selbständig ermittelt werden, dies bedeutet aber, dass das XML-Dokument gleich als erstes über die Anwendung angegeben werden müsste.

Eine andere Möglichkeit besteht darin, über die Oberfläche den Namen des Root-Elements mit anzugeben. Noch ein anderer Weg wäre die Festlegung des Root-Elements über die Konfigurationsdatei. Der beste Weg ist das Ermitteln anhand des mit angegebenen XML-Dokuments. Ein weiterer Eintrag in der Konfigurationsdatei brächte den Nachteil mit sich, dass der Anwender jedesmal beim Wechsel seines XML-Dokuments den Namen des Root-Elements in der Konfigurationsdatei ändern müsste. Ein Nachteil der Eingabe des Root-Elements über die Oberfläche wäre, dass ebenso beim Wechsel des XML-Dokuments zu der jeweiligen Registerkarte gesprungen werden müsste, um den Namen dort einzutragen. Das Template beinhaltet das FO-Element `fo:root`, welches das eigentliche Root-Element des FO-Dokuments ist. Hier müssen die Master-Pages, sowie die Seiteninhalte verarbeitet werden. Um die Master-Pages zu erzeugen, kann die Methode aus dem XSLT-Stylesheet zur Generierung der jeweiligen Master-Pages aufgerufen werden. Diese liefert dann das `fo:layout-master-set` Element zurück, welches die jeweiligen Master-Pages aus der dazugehörigen Konfigurationsdatei ausliest. Ein kurzes Beispiel zeigt, wie dies aussehen könnte.

```
<xsl:template match="DocumentRoot">
    <fo:root>
        <xsl:call-template name="createLayoutMasterSets" />
        ....
    </fo:root>
</xsl:template>
```

Fig. 6-21 : Template Match mit Generierung der Master-Pages

Damit der Anwender nun auch die Inhalte seiner Seiten erzeugen kann, werden die in der Konfigurationsdatei gesetzten Templates dynamisch erzeugt. Dabei spielt hier die Reihenfolge aus dem XML-Dokument der Seitensequenzen eine große Rolle. So wie diese angeordnet sind, werden die Seiten ebenfalls angelegt

und ausgegeben. Um diese Methoden nun zu erzeugen, legt die Anwendung in dem Stylesheet pro Eintrag Templates an, die über das obige Root-Element, nach dem die Master-Pages erzeugt wurden, nacheinander aufgerufen werden. Diese Aufrufe würden folgendermaßen aussehen.

```
<xsl:template match="DocumentRoot">
    <fo:root>
        ....
        <xsl:call-template name="fot_page_sqn_Deckblatt"/>
        <xsl:call-template name="fot_page_sqn_IVZ"/>
    </fo:root>
</xsl:template>
```

Fig. 6-22 : Template Match mit Aufruf der Template-Methoden

Damit diese Templates auch aufgerufen werden können, wird das Gerüst hierfür angelegt. Innerhalb dieser Templates werden nun die Seitensequenzen erzeugt. Um die jeweiligen Seitensequenzen zu erzeugen, wird anhand des Template-Namens der Eintrag aus dem XML-Dokument bezogen und dieser dann an die Methode des XSL-Stylesheets zur Generierung der Seitensequenzen übergeben. Somit wird pro Template genau eine Seitensequenz erzeugt. Die Struktur der Templates ist dabei immer gleich, was man anhand der folgenden Abbildung erkennen kann.

```
<xsl:template name="fot_page_sqn_Deckblatt">
    <fo:page-sequence master-reference="Deckblatt">
        <xsl:call-template name="generateStaticContents">
            <xsl:with-param name="Section">Deckblatt</xsl:with-param>
        </xsl:call-template>
        <fo:flow flow-name="xsl-region-body">
            ....
        </fo:flow>
    </fo:page-sequence>
</xsl:template>
```

Fig. 6-23 : Template mit Erzeugung der Seitensequenzen

Der Master für die zu erzeugende Seite wird anhand der Zuweisung aus dem XML-Dokument mit den Konfigurationen der Master-Pages ermittelt. Hier steht jede Referenz für die Seitensequenzen betreffend des verwendeten Masters. Somit kann der Master über das Attribut `master-reference` bezogen werden. Darauf werden die statischen Inhalte erzeugt. Die Template-Methode *generateStaticContents* bekommt als Parameter den Namen des Templates mit überliefert, um über diesen die Informationen der Inhalte aus dem XML-Dokument zu beziehen. Nachdem die statischen Inhalte erzeugt wurden, wird der Hauptbereich der Seite gefüllt. Über die Anwendung hat der Anwender das Template angegeben, welches hierbei den Inhalt der Seiten beschreibt. Dieses Template liegt in seinem XSL-FO Stylesheet, welches in dem erzeugten Stylesheet mit eingebunden wurde, damit die Templates für die Inhalte referenziert werden können. Somit wird eine enge Verbindung zwischen diesen beiden Stylesheets geschaffen.

6.1.10 Konfigurationsdatei zur XSL-FO Entwicklungsumgebung

Damit das Ergebnisdokument ausgespielt werden kann, wird eine Konfigurationsdatei verwendet. Diese beinhaltet Parameter, die vom Anwender gepflegt werden sollten. Es kann sein, dass der Anwender mehrere FO-Prozessoren auf seinem System installiert hat. Somit sollte es möglich sein, diese in der Konfigurationsdatei mit aufzunehmen. Hierzu sollte der Pfad zum jeweiligen Prozessor mit angegeben werden, damit diese Datei über die Anwendung zur Vorschau des PDF-Dokuments angesprochen werden kann. Ebenso muss jeder Prozessor so konfiguriert werden können, dass dessen Übergabeparameter flexibel ausgetauscht werden können. Jeder FO-Prozessor hat seine eigenen Parameter zur Bestimmung des FO-Dokuments, sowie des PDF-Dokuments. Zum Beispiel unterscheiden sich die Kommandos bei Antenna House XSL Formatter und beim Apache FOP. Somit ist es notwendig für jeden FO-Prozessor weitere Variablen aufzunehmen, die vom Anwender gepflegt werden müssen. Der Eintrag für diese Konfigurationsdatei kann folgendermaßen aussehen.

```
[FO_Processors]
Processors = Antenna, Fop, Xep

[Antenna]
Title        = Antenna House Formatter
Directory    = D:\Programme\XSLFormatterV42\XSLCmd.exe
Command_Line = -d %1 -o %2
Config_dir   =
```

Fig. 6-24 : Konfigurationsdatei mit Eintrag für FO-Prozessoren

Das Feld *FO_Processors* beinhaltet die Variable *Processors*, in der die unterschiedlichen FO-Prozessoren durch Komma getrennt eingetragen werden können. Diese Namensgebung ist nachher die Verbindung zu den Daten des Prozessors. Im obigen Beispiel gibt es den Eintrag *Antenna*. Über diesen Eintrag wird zum Feld *Antenna* gesprungen, welches Informationen über den FO-Prozessor enthält. Hierzu ist es wichtig, dass der Anwender einen Titel für den Prozessor angeben kann. Dieser Titel erscheint dann bei der Auswahl des FO-Prozessors bei der Erstellung der Vorschau. Somit kann der Anwender selbst den Namen vergeben, wie er später über die Oberfläche angezeigt werden soll. Dabei sollte darauf geachtet werden, dass der Titel für den FO-Prozessor eindeutig bleibt, da diese Daten dann von der Anwendung ausgewertet werden müssen. Wenn der Anwender zweimal den selben Titel für unterschiedliche Prozessoren verwenden würde, so könnte es zu Fehlern der Anwendung kommen, da nicht klar definiert wurde, welcher FO-Prozessor verwendet werden soll.

Damit der FO-Prozessor angesprochen werden kann, sollte dem System der Pfad zu dieser Anwendung mitgeteilt werden können. Hierzu gibt der Anwender über eine Variable, bspw. *Directory*, den kompletten Pfad zum Prozessor an. Desweiteren sollten nun die unterschiedlichen Übergabeparameter mit angegeben werden können. Der geschickteste Weg wäre eine Variable, im oberen Beispiel *Command_Line*, in der nun der Kommandozeilenaufruf des FO-Prozessors enthalten ist. Dabei gibt es drei zu vergebene Zeichenfolgen, die jeweils mit dem Prozentzeichen beginnen. *%1* wäre dabei für den Platzhalter des Ergebnisdokuments gesetzt, welches über die Oberfläche angegeben werden kann. *%2* wäre für den Platzhalter des PDF-Dokuments reserviert, welches vom System aus dynamisch erzeugt wird. Optional gibt es dann noch den Platzhalter *%3*, welcher beim Einsatz des FO-Prozessors Apache FOP zum Tragen kommt. Hier kann eine Konfigurationsdatei für den Support von Schriftarten mit angegeben werden, welche über die Variable *Config_dir* gesetzt werden könnte. Mit diesen Variablen wären jegliche Einstellungen bezüglich der Ausführung des FO-

Prozessors gegeben. Eine weitere Möglichkeit, wie man die Problematik der Übergabeparameter regeln könnte ist, für jeden dieser Parameter eine eigene Variable zu definieren, die dann nur noch den jeweiligen Parameter aus der Kommandozeile, bspw. -d, enthält.

Für die Transformation des FO-Dokuments können bestimmte XSLT-Prozessoren verwendet werden. Genauso wie die FO-Prozessoren sollten hier ebenso gewisse Einstellungen vorgenommen werden können. Der Anwender sollte so mehrere XSLT-Prozessoren angeben können, die auf seinem System installiert sind, sowie auswählen können, über welchen Prozessor das FO-Dokument erstellt werden soll. Somit kann der Anwender jedesmal in der Konfigurationsdatei den XSLT-Prozessor austauschen, um so mehr Flexibilität in den Ablauf zu bringen. Er ist somit gleichzeitig nicht auf die Einstellungen der Anwendung beschränkt, sondern kann selbst bestimmen, welche Einstellungen er diesbezüglich treffen möchte. Diese Einträge könnten wie folgt aussehen.

```
[XSLT_Processors]
UseProcessor = Xalan

[Xalan]
Command_Line = java -cp xslt.jar;xalan.jar
               org.apache.xalan.xslt.Process
               -xsl %1 -in %2 -out %3
```

Fig. 6-25 : Konfigurationsdatei mit Eintrag für XSLT-Prozessoren

Unter der Variablen *UseProcessor* wird derjenige XSLT-Prozessor angegeben, über den das FO-Dokument erstellt werden soll. Über den Namen des Prozessors kann dann zu den einzelnen Feldern gesprungen werden, in denen Informationen bezüglich ihres Aufrufs gegeben sind. Hierzu gibt es die Variable *Command_Line*. Ein Beispiel soll zeigen, wie diese ausgewertet werden kann. Wenn Xalan verwendet werden soll, so bietet sich der obige Inhalt als eine Möglichkeit an.

Wie man sehen kann, gibt es auch hier wieder bestimmte Platzhalter, angeführt mit einem Prozentzeichen. Die Werte, die jeweils mit einem Prozentzeichen gesetzt wurden, werden durch die Anwendung ersetzt und die Kommandozeile über die Anwendung ausgeführt. Hierzu kann auch die gesamte Kommandozeile samt Prozentzeichen angegeben werden, damit diese nur noch ausgeführt werden soll. Das bedeutet, dass hier die jeweiligen Jar-Dateien mitangegeben werden können. *%1* würde für das XSLT-Stylesheet stehen, welches dabei für das Erzeugen des FO-Dokuments sorgen würde. *%2* steht für den Platzhalter des einzulesenden XML-Dokuments, welches den Inhalt der Seiten darstellt. Der letzte Platzhalter gibt an, wo das FO-Dokument gespeichert werden soll. Dieser Parameter wird über die Oberfläche durch den Anwender eingegeben. Diese Zeile lässt sich natürlich auch verkürzen, wenn man die Jar-Datein weglässt. Diese könnten dann in den Umgebungsvariablen fest definiert werden.

Natürlich kann man sich auch hier eine andere Möglichkeit einräumen, indem man für die drei Eingabeparameter wiederum eigene Variablen verwenden würde. Jedoch wäre hier der programmiertechnische Aufwand erhöht, da erst jede dieser Variablen ausgewertet werden müsste, um so die Kommandozeile zusammenzusetzen. Durch die obere Lösung gäbe es nur einen Aufruf und somit hätte man einen Wert, in dem nur noch die Platzhalter ersetzt werden müssten.

Damit die benötigten Stylesheets von der Anwendung gefunden werden können, sollte es ein Feld für deren Verzeichnisse geben, die vom Anwender konfiguriert werden können. Benötigte Stylesheets sind vor allem die XSLT-Stylesheets, die die Master-Pages und Seitensequenzen erzeugen. Somit bleibt die Anwendung

flexibel und kann sich an die Verzeichnisstruktur des Anwenders anpassen. Hier ein Beispiel, wie dies in der Konfigurationsdatei aussehen könnte.

```
[Directories]
MasterPagesStylesheet_dir    = D:\StudioXFO\FoT_Gen_MasterPages.xsl
PageSequencesStylesheet_dir = D:\StudioXFO\FoT_Gen_PageSequences.xsl
GeneratedStylesheet_dir      = C:\tmp\project_FoT\xsl
Batch_dir                    = C:\tmp\project_FoT\bat
Preview_dir                  = C:\tmp\project_FoT\pdf
```

Fig. 6-26 : Konfigurationsdatei mit Eintrag für Verzeichnisse

Der Eintrag *Directories* beinhaltet alle Verweise auf die benötigten Stylesheets, welche zur Verarbeitung der Master-Pages und Seitensequenzen benötigt werden, sowie auf Verzeichnisse, in denen von der Anwendung erzeugte Dateien abgelegt werden. Die beiden Variablen *MasterPagesStylesheet_dir* und *PageSequencesStylesheet_dir* geben hier jeweils die Pfade zu den XSLT-Stylesheets für die Auswertung der Konfigurationsdateien der Master-Pages und Seitensequenzen an.

Wie bereits aus den vorigen Kapiteln bekannt, werden bestimmte Dateien von der Anwendung generiert. Diese betreffen die beiden Batch-Datein für die Erzeugung des FO-Dokuments sowie des PDF-Dokuments, sowie das XSL-FO Stylesheet, welches das XML-Dokument des Anwenders parst und FO daraus erzeugt. Hierzu bieten sich drei Variablen an, im oberen Beispiel betrifft dies *GeneratedStylesheet_dir* für die Angabe eines Verzeichnisses, in dem die erzeugten XSL-FO Stylesheets kopiert werden sollten, *Batch_dir* und *Preview_dir*. für die Erzeugung der Batch- und Previewdateien. Somit lassen sich auch diese Einstellungen vom Anwender vornehmen und müssen nicht fest von der Anwendung aus implementiert werden.

6.1.11 Generierung des FO- und PDF-Dokuments

In diesem Teil der Diplomarbeit sollen die beiden wichtigsten Prozessoren beschrieben werden, die für die jeweilige Generierung des FO-Dokuments, sowie der Erzeugung des PDF-Dokuments verantwortlich sind.

Erzeugung des FO-Dokuments

Damit das FO-Dokument erzeugt werden kann, wird ein sogenannter XSLT-Prozessor verwendet. Dieser Prozessor bekommt als Übergabeparameter zum einen das von der Anwendung erzeugte XSL-FO Stylesheet sowie das vom Anwender erstellte XML-Dokument (Inhalte für die PFD-Seiten) übergeben. Der XSLT-Prozessor führt die Inhaltstemplates des XSL-FO Stylesheets aus und gibt das entsprechende Ergebnisdokument weiter. Der Aufruf des XSLT-Prozessors wird über die Konsole erfolgen. Ein Beispiel soll zeigen, wie dieser Aufruf aussehen könnte.

```
java org.apache.xalan.xslt.Process
     -xsl obj_fot-14112007.xsl
     -in  export.xml
     -out preview_fot-14112007.fo
```

Fig. 6-27 : Kommandozeilenaufruf mit Xalan

Dieser Aufruf wurde mit dem XSLT-Prozessor Xalan ausgeführt. Die wichtigsten Merkmale hier sind die Übergabeparameter, gekennzeichnet durch die Bezeichnungen *-xsl*, *-in* und *-out*. Hierbei werden ihm über *-xsl* das generierte XSL-FO Stylesheet, über *-in* das XML-Dokument sowie über *-out* das Ergebnisdokument übergeben. Wenn alles ordnungsgemäß und sauber abgelaufen ist, wurde die Datei .fo erstellt. Falls Fehler aufgetreten sind, werden diese über die Konsole ausgegeben. Somit kann nachvollzogen werden, an welcher Stelle der Fehler aufgetreten ist, um diesen dann im XSL-FO oder FO-Dokument betrachten und verbessern zu können. Die Einstellungen für den XSLT-Prozessor findet der Anwender in der Konfigurationsdatei.

Erzeugung des PDF-Dokuments

Das erstellte FO-Dokument über den XSLT-Prozessor kann nun über den FO-Prozessor in das Format PDF gebracht werden. Welcher FO-Prozessor dabei verwendet werden soll, legt der Anwender über die Oberfläche fest. Die Einstellungen zu diesem FO-Prozessor können über die Konfigurationsdatei vom Anwender festgelegt werden. Der Aufruf eines FO-Prozessors zeigt das folgende Beispiel.

```
fop.bat -c   userconfig.xml
        -fo  preview_fot-14112007.fo
        -pdf preview_obj_fot-14112007.pdf
```

Fig. 6-28 : Kommandozeilenaufruf mit FOP

Hier wurde der FO-Prozessor Apache FOP aufgerufen. Bei den FO-Prozessoren gibt es zwei wesentliche Übergabeparameter, die sich in ihrer Bezeichnung unterscheiden. Zum einen muss der Prozessor das FO-Dokument kennen, welches hier über den Aufrufparameter *-fo* geschieht. Zum anderen muss der Prozessor wissen, in welches Format und wohin das Ergebnisdokument gespeichert werden soll. Dies geschieht mit dem Aufrufparameter *-pdf.* Hier wird nun aus dem FO-Dokument ein PDF-Dokument generiert. Falls Fehler aufgetreten sind, werden diese über die Oberfläche, mit einer entsprechenden Fehlerquelle gekennzeichnet, ausgegeben.

6.2 Realisierung der XSL-FO Entwicklungsumgebung

In diesem Teil der Diplomarbeit wird näher auf die Implementierung der Anwendung eingegangen, was die Oberfläche und die dahinterliegenden Klassen betrifft. Es soll erkannt weden, welcher Ansatz sich hierzu eignet, und wie dieser am besten umgesetzt werden kann. Neben der Implementierung soll die Anwendung nach diesem Vorgang ausführlichst getestet werden, um die genannten Anforderungen zu erfüllen. Die Anwendung ist durch die mit .NET gegebenen Möglichkeiten nach dem Model-View-Controller Prinzip aufgebaut. Das wesentliche Ziel des sogenannten MCV-Musters (Model-View-Controller) liegt in der Trennung der Präsentations- von der Anwendungslogik. Diese strikte Trennung hat den Vorteil, dass sich die einzelnen Komponenten flexibel austauschen lassen. Das MVC-Muster kapselt sich in drei Komponenten. Die Anwendungslogik liegt im Model, die Benutzeroberfläche wird durch die View dargestellt und der Controller interagiert zwischen Model und Controller. Die nachfolgende Abbildung spiegelt dieses Prinzip wieder.

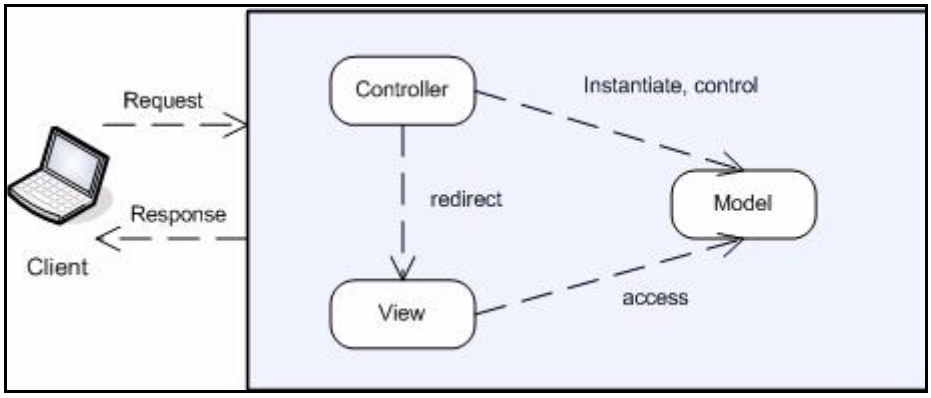


Fig. 6-29 : MVC-Modell nach [WAES06]

In dieser Architektur hat jede Komponente ihre eigene Funktionsweise. Das Model hat die Aufgabe, Daten, die durch den Anwender eingegeben werden, entsprechend zu verarbeiten. Das Model kennt seine Views nicht. Es empfängt nur die eingegebenen Daten und kann über eine Update-Methode die Views dazu veranlassen, geänderte Daten auf der Oberfläche zu aktualisieren. Das Model kann auch die Daten verwalten. Somit können diese gleichzeitig über das Model abgerufen werden. Zur Repräsentation der Daten dient die View. Sie stellt die Benutzeroberfläche der Anwendung dar und beinhaltet keine Anwendungslogik. Sie ist implizit nur da, um die Daten darzustellen.

Der Controller nimmt Eingaben des Anwenders entgegen und ruft gewünschte Aktionen über das Model auf. Im obigen Beispiel werden die Eingaben des Anwenders durch den Request dargestellt. Dieser Request kann ein Klick auf einen Button oder eine Tastatureingabe sein. Diese Benutzereingaben werden vom Controller erfasst und an das Model weitergeleitet.

Dieser Prozess soll anhand der Erstellung einer Master-Page verdeutlicht werden. Der Anwender ruft über die Oberfläche das Menü zur Erstellung einer Master-Page auf. Der Controller fängt die Eingabe ab und öffnet die entsprechende View. Die View initialisiert ihre Oberfläche mit leeren Daten. Der Anwender gibt nun die Daten über die View ein und betätigt den Speicher-Button. Der Controller reagiert auf dieses Ereignis. Die Daten werden aus der Oberfläche entnommen und an das Model übergeben. Das Model nimmt die Daten auf und reagiert auf das vom Controller abgefangene Event, in dem es die passende Methode aufruft. Das Model speichert nun die Daten in einem XML-Dokument ab und sendet dem Controller den Erfolg. Der Controller kann nun darauf reagieren und eine entsprechende Erfolgsmeldung der View anfordern. Somit bekommt der Benut-

zer diese Erfolgsmeldung mitgeteilt. Da die Anwendung mit der Programmiersprache C# in Microsoft Visual Studio .Net geschrieben wurde, ist dieses Prinzip schon mit inbegriffen. Das bedeutet, wenn der Anwender auf einen Button klickt oder in der Menüleiste eine Aktion ausführen möchte, so wird hierzu schon bereits eine Methode angelegt, die dann nur noch vom Entwickler ausprogrammiert werden muss. Aus Zeitgründen wurde dieser Ansatz gewählt, da es eine Vielzahl an Events abzufangen gäbe, die durch diese Möglichkeit einfacher zu lösen sind. Im nachfolgenden Kapitel, soll etwas näher auf die einzelnen Klassen eingegangen werden, damit die Architektur der Anwendung samt Entwurfsmuster verständlich wird. In den folgenden Kapiteln wird nun näher auf die einzelnen Klassen der Anwendung eingegangen, um die Zusammenhänge und Entwurfsmuster, die dabei eingesetzt wurden, zu verstehen. Dabei werden unterschiedliche Muster verwendet, um die Anwendung so modular und wiederverwendbar zu halten.

6.2.1 Trennung der Repräsentations- und Anwendungslogik

.NET bietet schon die Möglichkeit, auf das Model-View-Controller Prinzip aufzubauen. Es wird die einfache Erstellung von Views mittels .NET in Betracht gezogen, um die einzelnen Benutzeroberflächen der Anwendung zu implementieren. Die folgende Abbildung zeigt die wesentlichen Klassen, die für die einzelnen Ansichten der Anwendung verwendet werden.

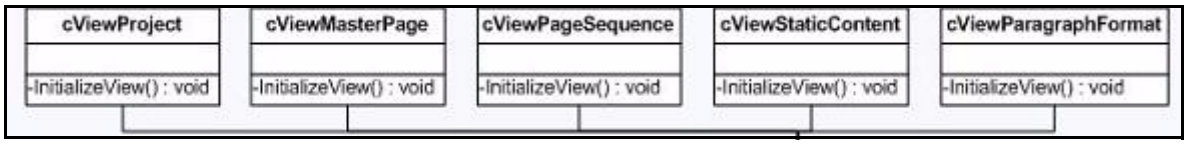


Fig. 6-30 : Die Viewing-Klassen zur Darstellung der Benutzeroberfläche

Für die Darstellung der Projekteinstellungen sorgt die Klasse `cViewProject`. Von dieser Oberfläche aus kann zu den einzelnen Bereichen Master-Pages, Seitensequenzen und Absatzformate gesprungen werden, um diese bearbeiten zu können. Diese Klasse stellt die eigentliche Projektoberfläche dar. Jegliche Aktionen gehen von dieser Klasse aus. Über die Methode `InitializeView` werden alle Daten aus den Konfigurationsdateien ermittelt (Zugriff über die Anwendungslogik) und über die Oberfläche ausgegeben. Alle für das Projekt definierten Master-Pages, Seitensequenzen und Absatzformate werden anhand den Konfigurationen ermittelt und initialisiert. Somit hat der Anwender nun Zugriff auf diese Objekte und kann diese entsprechend verändern, entfernen oder neue Objekte erstellen, um diese dem Projekt hinzuzufügen.

Für die Darstellung der Master-Pages sorgt die Klasse `cViewMasterPage`. Hier werden alle Einstellungen bezüglich einer Master-Page getroffen bzw. können bearbeitet werden. Je nachdem, ob der Anwender eine vorhandene Master-Page bearbeiten oder neu anlegen möchte, wird die Methode `InitializeView` aufgerufen. Bei Neuanlegen einer Master-Page bekommt diese keine Werte mit, und bietet so ihre Oberfläche mit leerem Inhalt an, wobei einige Werte bereits vorbelegt werden, um die Eingabe für den Anwender zu erleichtern. Wenn der Anwender eine Master-Page ausgewählt hat, so bekommt die View die jeweiligen Daten übergeben, und stellt sie über diese Methode an der Oberfläche dar. Da die Master-Pages bereits beim Laden des Projekts ermittelt werden, können so die Daten der ausgewählten Master-Page über das instanzierte Objekt bezogen und dargestellt werden.

Für die Darstellung der Seitensequenzen sorgt die Klasse `cViewPageSequence`. Hier regelt die Darstellung der Daten ebenso die Methode `InitializeView`. Diese View hat eine Verbindung zur Klasse `cViewStaticContent`, welche die Daten der statischen Inhalte darstellt. Hier können nun Inhalte in den einzelnen Berei-

chen einer Seite definiert werden, ebenso wie die Auswahl eines Inhaltstemplates aus dem XSL-FO Stylesheet getroffen werden, um dem Druckbereich einen Inhalt zu geben. Im Verlauf dieser Diplomarbeit werden diese Einstellungen anhand der Benutzeroberfläche deutlicher dargestellt.

Für das Anlegen und Bearbeiten von Absatzformaten wird die Klasse `cViewParagraphFormat` verwendet. Wie bei den anderen Klassen auch gibt es die Methode `InitializeView` die die entsprechenden Daten an der Oberfläche dem Anwender darstellt. Jede dieser Klassen hat eine Beziehung zum Model, welches die Anwendungslogik implementiert. Dies bewirkt die strikte Trennung der Präsentations- und Anwendungslogik. Sollten die Methoden, die hinter den Events liegen, geändert werden, so ist die Auslagerung einer Klasse, welche die Anwendungslogik implementiert, ideal, da hier die Funktionen separat angepasst werden können, ohne dabei auf die Oberfläche der Anwendung Einfluss zu nehmen. Die Klasse kann somit jederzeit ausgetauscht werden. Die verschiedenen Klassen der View sollten nur wissen, welche Methoden aufgerufen werden müssen. Die folgende Abbildung repräsentiert die Anwendungslogik.

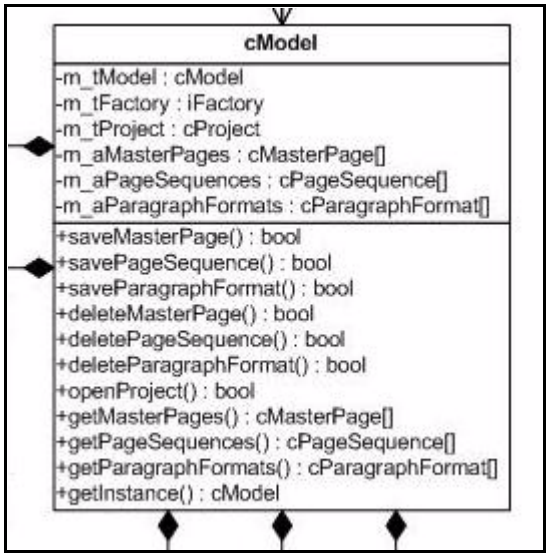


Fig. 6-31 : Die Klasse cModel, Implementierung der Anwendungslogik

Die Klasse `cModel` stellt das Model dar. Hier werden die vom Anwender in der Oberfläche eingegebenen Daten an die entsprechenden Methoden übergeben. Die Klasse stellt Methoden bezüglich dem Speichern, Entfernen und Bearbeiten von Master-Pages, Seitensequenzen und Absatzformaten zur Verfügung. Auch neu angelegte Projekte regelt das Model. Das Model enthält Referenzen auf die jeweiligen im System vorhandenen Master-Pages, Seitensequenzen und Absatzformate. Beim Öffnen eines Projekts werden die jeweiligen Daten aus den Konfigurationsdateien ausgelesen und entsprechend über das Model initialisiert.

Das Model wurde nach dem Singelton-Pattern implementiert. Das bedeutet, dass diese Klasse nur einmal initialisiert werden kann. Somit können nicht mehrere Instanzen des Models erzeugt werden. Das hat den Vorteil, dass, wenn die Anwendung gestartet und ein Projekt geöffnet wird, eine Instanz des Models erzeugt wird, die dann die bereits vorhandenen Master-Pages, Seitensequenzen und Absatzformate des Projekts erzeugt hat. Somit kann immer auf der aktuellsten Konfiguration gearbeitet werden. Sollte ein neues Projekt angelegt werden, so wird hier ebenfalls eine Instanz des Models erzeugt und die Werte alle mit *null* initialisiert. Somit können diese dann über die Anwendung gesetzt und erzeugt werden.

Der Vorteil dieser Architektur liegt in der Trennung der Repräsentations- und Anwendungslogik. Das Model kennt seine Views nicht. Die verschiedenen Views je-

doch kennen das Model. Wenn sie initialisiert werden, erstellen sie jeweils eine Instanz des Models. Somit haben sie Zugriff auf dessen Methoden. Diese Methoden können dann je nach Benutzereingaben aufgerufen werden. Die Ergebnisse des Models werden dann wieder über die Oberfläche dargestellt. Dies ist auch ein wesentlicher Gesichtspunkt, warum sich das Model-View-Controller-Entwurfsmuster für solche Anwendungen optimal einsetzten lässt.

6.2.2 Fabrikmethoden zur Erzeugung der Klassen

Damit das Model nicht immer die jeweiligen Klassen für die Master-Pages, Seitensequenzen und Absatzformate instanzieren muss, wurde ein Interface für die Erzeugung dieser Klassen bereitgestellt.

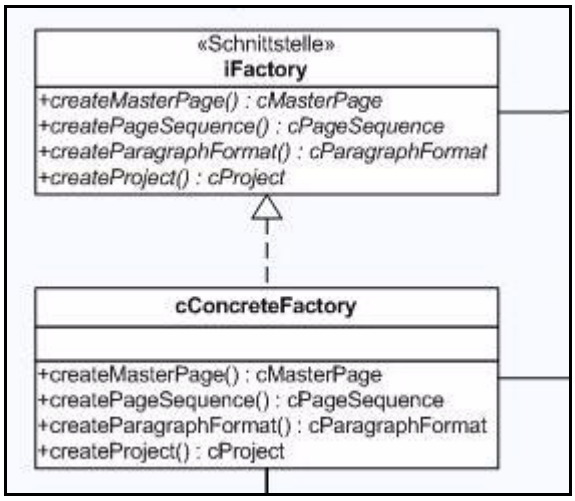


Fig. 6-32 : Die Schnittstelle iFactory, Fabrikmethoden

Das Interface `iFactory` ist als Member-Variable in der Klasse `cModel` integriert. Dieses wird dann mit der Klasse `cConcreteFactory` initialisiert. Die konkrete Klasse implementiert die Methoden des Interfaces. Diese Methoden erzeugen Instanzen der Klassen `cMasterPage` mit der Methode `createMasterPage`, `cPageSequence` mit der Methode `createPageSequence` und `cParagraphFormat` mit der Methode `createParagraphFormat`. Die konkrete Klasse kann somit ausgetauscht werden, wenn sich etwas bei der Instanzierung der Klassen ändern würde. Das System selbst sollte unabhängig davon sein, wie die einzelnen Master-Pages, Seitensequenzen und Absatzformate repräsentiert bzw. erzeugt werden. Eine Klassenbibliothek bietet sich in diesem Falle an, da die Implementierung der verschiedenen Objekte nicht nach außen hin dargestellt, sondern nur die Schnittstelle bekannt gemacht werden soll. [GAMMA96]

Bei diesen Anforderungen liegt es nahe, eine Fabrik zu verwenden. Es bieten sich jedoch noch einige Vorteile des Musters. Die Klasse `cConcreteFactory` ist verantwortlich für die Erzeugung der verschiedenen Objekte und trennt somit den Client von diesen Implementierungsklassen. Sollten, bspw. bei der Bearbeitung dieser Objekte, Veränderungen auftreten bzw. Methoden dieser Objekte aufgerufen werden, so manipuliert der Client diese Objekte durch die jeweilige Schnittstelle. Ein weiterer Vorteil, der sich hierbei noch bietet ist, dass die Fabrikklasse, welches das Interface `iFactory` implementiert, einfach ausgestauscht werden kann. Da die Klasse die jeweiligen Objekte erzeugt, gibt es einen Komplettaustauch der gesamten Objekte.

Der einzige Nachteil, der bei dem Einsatz eines solchen Musters entstehen kann, ist die Abhängigkeit der zu implementierenden Methoden. Jede Methode, die in dem obigen Interface angegeben wurde, muss von den jeweiligen Klassen implementiert werden. Sollten neue Objekte in der Klassenbibliothek aufgenommen werden, so muss die Schnittstelle dementsprechend erweitert werden. [GAMMA96]

6.2.3 Klassenansicht der Master-Pages

Die Klasse `cMasterPage` stellt eine Master-Page dar. Hier können sämtliche Daten, die über die Anwendung gesetzt werden über die "get" und "set"-Methoden bezogen bzw. verändert werden. Die Klasse besitzt Member-Variablen bezüglich des Namens, seitlichen Abständen, Seitenformat und enthält Referenzen zu den fünf Bereichen Druckbereich, Kopfbereich, Fußbereich, linker und rechter Seitenbereich.

Wie bereits aus der Analyse bekannt, kann eine Seite in die genannten Bereiche aufgeteilt werden. Man spricht dabei von einer *Ist-Teil-Von* Beziehung. Hierbei bietet sich die feste Komposition an. Das bedeutet, jeder dieser Bereiche wird nochmals als eigenständige Klasse dargestellt und als Member-Variable der Klasse `cMasterPage` hinzugefügt. Somit sind sie feste Bestandteile dieser Klasse. Dabei kann eine Master-Page immer nur einen Bereich des jeweiligen Typs beinhalten, sprich es gibt genau einen Druckbereich, eine Kopfzeile, eine Fußzeile, einen linken sowie einen rechten Seitenbereich. Für die Kapselung der Daten des Druckbereichs wurde die Klasse `cRegionBody`, für die Daten des Kopf- und Fußbereichs die Klassen `cRegionBefore` und `cRegionAfter`, für die Daten des linken und rechten Seitenbereichs die Klassen `cRegionStart` und `cRegionEnd` implementiert.

Diese Klassen haben wiederum nur "get" und "set"-Methoden, um die jeweiligen Daten anzulegen, bzw. zu verändern. Die folgende Abbildung zeigt nochmals deutlich die feste Komposition der Klasse `cMasterPage`.

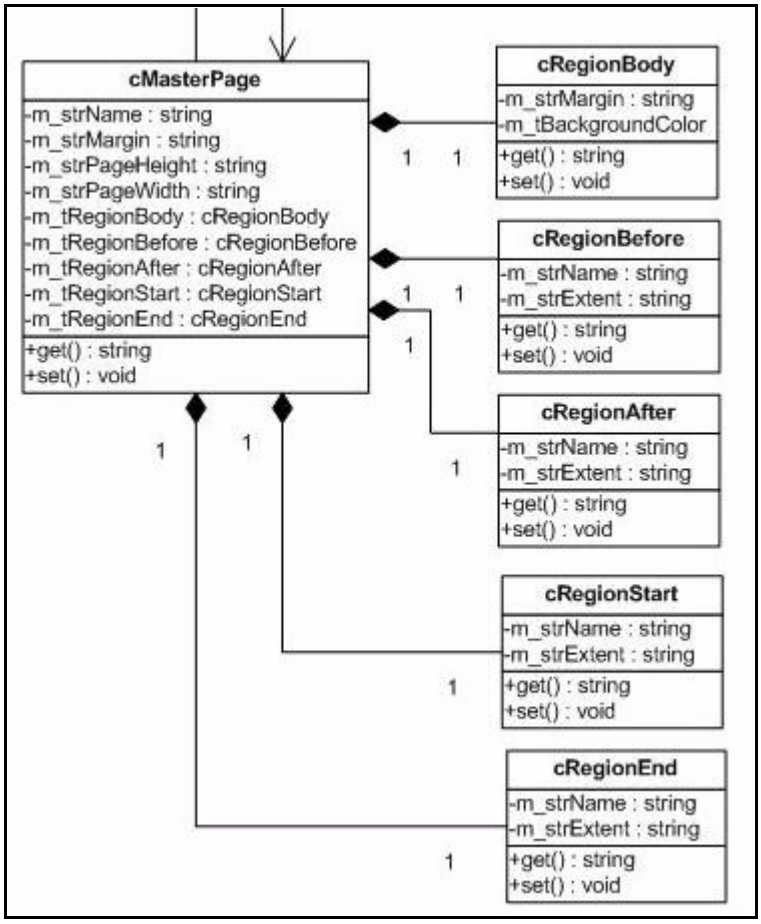


Fig. 6-33 : Die Klasse cMasterPage, Einstellungen einer Master-Page

6.2.4 Klassenansicht der Seitensequenzen

Die Klasse `cPageSequence` stellt eine Seitensequenz dar. Die Klasse kapselt jegliche Daten, die für das Erstellen und Bearbeiten einer Seitensequenz notwendig sind und von den FO-Prozessoren unterstützt werden. Da eine Seitensequenz unterschiedliche statische Inhalte besitzen kann, wurde diese Klasse, mit `cStaticContent` bezeichnet, ausgelagert. Da es mehrere statische Inhalte für eine Seitensequenz geben kann, bietet sich hier ein Array an, welches statische Inhalte beinhaltet. Diese Verbindung wird durch die Komposition zwischen `cPageSequence` und `cStaticContent` dargestellt. Einem statischen Inhalt können bestimmte Rahmeneinstellungen und Absatzformate zugeordnet werden. Diese Objekte wurden in extra Klassen gekapselt und als Komposition der Klasse `cStaticContent` implementiert.

In vorherigen Kapiteln wurde die Idee gefasst, dass bspw. bei einer Kopfzeile horizontale, sowie vertikale Linien verwendet werden können. Diese Linien sind alle vom Typ `cBorder`. Es gibt vier verschiedene Typen von Linien bzw. Designmöglichkeiten. Zum einen gibt es horizontale Linien, die sich oben und unten erschließen, sowie vertikale Linien auf den Ebenen links und rechts. Diese Designmöglichkeiten sind in einer Member-Variablen `m_strBorderType` zusammengefasst worden. Somit kann in der Klasse `cStaticContent` ein Array angelegt werden, welches diese Objekte enthält. Um herauszubekommen, wie diese Linie gezeichnet werden soll, wird das Array durchlaufen und anhand des entsprechenden Typs erkannt. Eine Linie besitzt dabei die Eigenschaften Linienstil, Linienstärke und Linienfarbe. Das Absatzformat wird im darauffolgenden Kapiteln noch näher betrachtet. Diese Klasse ist ebenso über eine Komposition mit

der Klasse `cStaticContent` verbunden. Somit kann jedem statischen Inhalt ein Absatzformat zugeordnet werden. Die untenstehende Abbildung verdeutlicht diese Entwürfe.

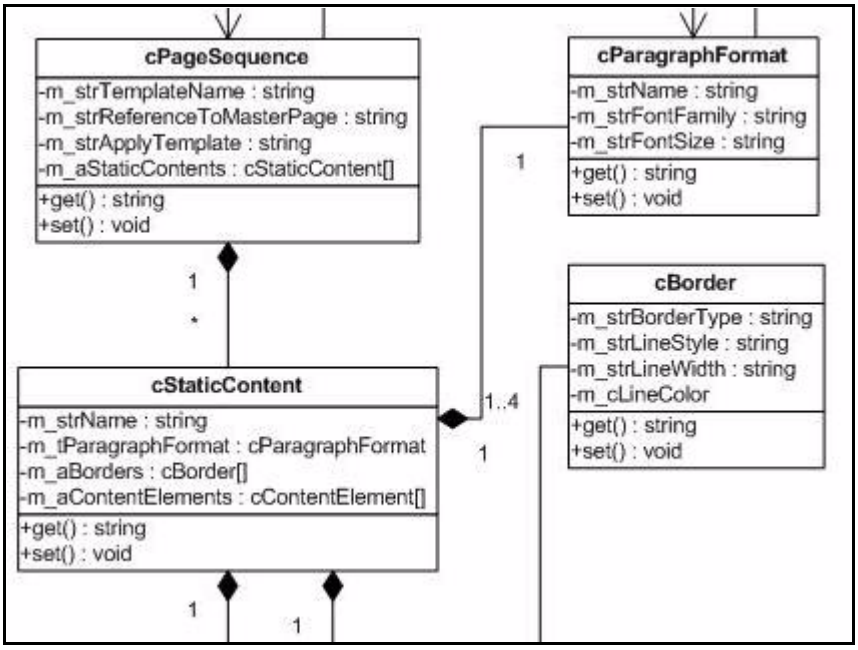


Fig. 6-34 : Die Klasse cPageSequence, Einstellungen der Seitensequenz

Für die Implementierung der Content-Elemente wurde eine abstrakte Klasse bereitgestellt, die drei Attribute zur Definition der Positionierung, Elementgröße und Hintergrundfarbe beinhaltet. Diese Attribute wurden den Unterklassen herausgezogen, da diese bei allen Klassen vorkamen. Die Klasse `cContentElement` stellt die beiden Methoden `getImage` und `createXMLNode` zur Verfügung und legt damit fest, dass sämtliche Klassen, die die Eigenschaften dieser Klasse erben, auch die beiden Methoden überschreiben müssen.

Für die verschiedenen Elemente in einem statischen Inhalt wurden Klassen implementiert, die die jeweiligen Daten kapseln. Für eine Seitenzahl wurde die Klasse `cPageNumber`, für eine Grafik die Klasse `cGraphic`, für eine Kapitelüberschrift die Klasse `cRetrieveMarker` und für einen simplen Content die Klasse `cContent` bereitgestellt. Diese Klassen erben die Eigenschaften der abstrakten Klasse `cContentElement`. Die beiden Methoden, die überschrieben werden müssen, sind `getImage` und `createXMLNode`. Durch `getImage` wird jedem Objekt eine Grafik zugeordnet, die beim Bearbeiten im jeweiligen Feld über die Anwendung angezeigt werden kann, um so dem Anwender darzustellen, welche Objekte an welchen Positionen des statischen Bereichs gesetzt worden sind. Durch `createXMLNode` wird der Inhalt des Objekts durch ein XML-Element dargestellt. Dies hat den Vorteil, dass das Model sich diese Informationen nicht aus dem Objekt direkt herausholen muss, sondern gleich den Knoten zurückgeliefert bekommt, ohne nun die jeweiligen Get-Methoden aufrufen zu müssen. Somit muss auch nicht mehr der Typ des Objekts abgefragt werden, sondern über eine einzige For-Schleife das Array `m_aContentElement` durchlaufen und die Methode `createXMLNode` aufgerufen werden. Diese Möglichkeit spart wiederum Code.

Für die statischen Elemente wurde ein Array angelegt, welches Objekte des Typs `cContentElement` besitzt. Bei diesem Entwurfsmuster handelt es sich um eine flexible Komposition. Der Vorteil, der sich bei einer solchen Implementierung bietet, ist, dass zur Laufzeit entschieden werden kann, ob der statische Inhalt Objekte der Content-Elemente `cPageNumber`, `cGraphic`, `cRetrieveMarker` oder `cContent` besitzt. Diese Objekte können dann in dem bereitgestellten Array der Klasse `cStaticContent` gesetzt werden. Ein weiterer Vorteil ist, dass diese Objekte alle vom selben Typ `cContentElement` sind und somit in einem Array gehalten werden können, ohne dafür Objekte in der übergeordneten Klasse anzulegen. Die untenstehende Abbildung zeigt diese Verbindungen.

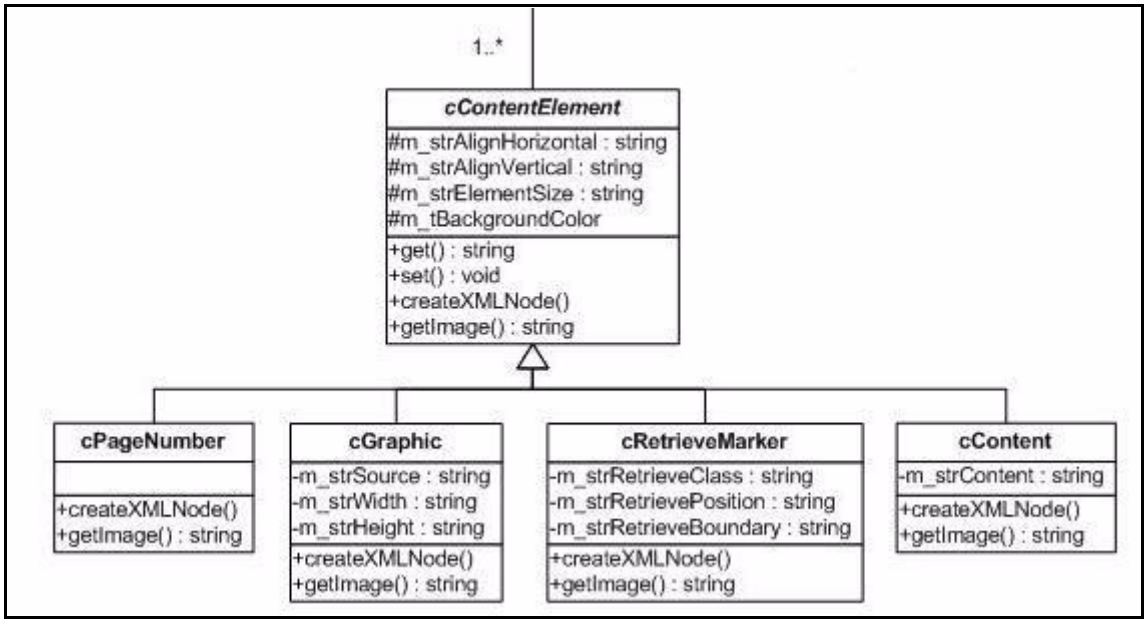


Fig. 6-35 : Die abstrakte Klasse cContentElement

6.2.5 Klassenansicht der Absatzformate

Die Klasse `cParagraphFormat` stellt ein Objekt eines Absatzformats dar. Hier werden sämtliche Daten bezüglich der Einstellung eines Absatzformats gekapselt. Die untere Abbildung zeigt einen kurzen Auszug der Klasse.

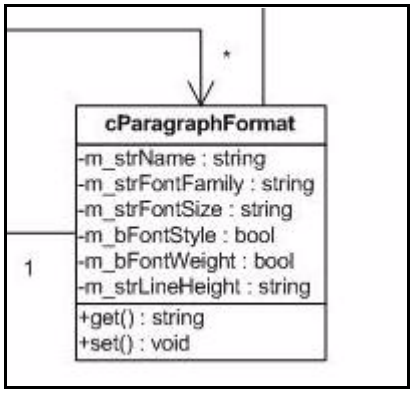


Fig. 6-36 : Die Klasse cParagraphFormat

Die Klasse hat eine *Ist-Teil-Von* Beziehung zur Klasse `cStaticContent`. Somit ist die Verbindung geschaffen, dass ein statischer Inhalt ein Absatzformat referenzieren kann, um dieses auf den Content anzuwenden. Ein Absatzformat besitzt alle Eigenschaften bezüglich der Schriftart, Schriftgröße, Zeilenabstände, usw. Diese können über die "set"-Methoden gesetzt werden. Für das Abrufen der jeweiligen Daten wurden "get"-Methoden implementiert.

7 Testdurchläufe der XSL-FO Entwicklungsumgebung

In dieser Ausarbeitung der Diplomarbeit sollen einige Testläufe mit Kundendokumenten durchgeführt werden, um zu sehen, ob das Arbeiten mit der Anwendung effizient durchgeführt werden und ob das gewünschte Ziel mit der Anwendung erreicht werden kann. Dazu wurden einige Dokumentationen betrachtet, wobei hier speziell auf das Seitenlayout und den Seiteninhalt eingegangen wurde.

7.1 Generierung von Instructioncards

Um zu testen, ob die Entwicklungsumgebung richtig eingesetzt werden kann, wurde dies mit Instructioncards getestet. Es handelt sich hierbei um eine Reihe von technischen Daten, die zusammengeheftet auf das Seitenformat DIN A6 gebracht wurden. Inhaltlich befinden sich Anweisungen und Informationen zu technischen Anlagen in diesen Dokumentationen. Diese haben ein ganz bestimmtes Design, was die Seitenvorlagen und Inhalte betrifft.

Die Seitenvorlagen konnten ohne Probleme definiert werden. Auch die Definition der Seitensequenzen und Zuordnung der jeweiligen Vorlage verlief ohne Probleme. Für die Darstellung der eigentlichen Seiteninhalte musste ein XSL-FO Stylesheet entwickelt werden, um zu testen, ob solche Instructioncards mit FO generierbar sind. Die einzige Schwierigkeit lag darin, unterschiedliche Master-Pages zu definieren, wo jeweils die verschiedenen Fußzeilen, diese unterscheiden sich in geraden und ungeraden Seitenfolgen, angepasst werden mussten.

Am rechten Seitenrand war ein sogenannter Thumb positioniert, der auf die jeweilige Kapitelnummer verwies, um beim Durchblättern darüber gleich zum jeweiligen Kapitel springen zu können. Mit etwas Erfahrung kommt man recht schnell zu dem erzielten Ergebnis. Jedoch ist eine Dokumentation sehr hilfreich dabei, um zu sehen, welche Einstellungsmöglichkeiten die Entwicklungsumgebung bietet.

7.2 Generierung von Produktkatalogen

Desweiteren, wurde getestet, ob sich Produktkataloge über die XSL-FO Entwicklungsumgebung ebenso einfach zusammenstellen lassen. Diese Produktkataloge beinhalten im allgemeinen ein Deckblatt, Inhaltsverzeichnis, Abbildungsverzeichnis, Glossar und Informationen zu verschiedensten Produkten, die von Firmen angeboten werden. Meistens sind dies technische Daten zu den einzelnen Produkten.

Die Definition der Seitenvorlagen konnte ohne Probleme durchgeführt werden. Auch die Anpassung der Seitensequenzen und Zuordnung der jeweiligen Vorlagen konnte ohne Probleme vollzogen werden. Da für die Seiteninhalte das jeweilige XSL-FO Inhaltstemplate aus dem Stylesheet ausgewählt werden kann, kommt man recht schnell zu einer passablen Vorschau und konnte so die restlichen Kleinigkeiten des Seitenlayouts anpassen. Durch das dynamische Erzeugen einer beliebigen Anzahl an Elementen in den Bereichen Kopf- und Fußzeile, sowie linkem und rechtem Seitenbereich ist es recht einfach gewesen, Elemente zu definieren und zu platzieren. Dies erleichtert den Austausch bzw. das Hinzufügen von weiteren Elementen.

7.3 Generierung von Technischen Dokumentationen

In einer Technischen Dokumentation befinden sich Bedienungs- oder Gebrauchsanweisungen zu einem Produkt. Ebenso gibt es Hinweise zu dem Produkt, wie auf sicherem Wege damit gearbeitet werden kann. [HETJ98] Für diesen Prozess wurde eine interne Dokumentation verwendet, um zu überprüfen, ob diese über die Anwendung mit XSL-FO erstellt werden kann.

Die Definition der Seitenvorlage konnte ohne Probleme durchgeführt werden. Das Seitenlayout der Technischen Dokumentation war recht einfach gegliedert. Es mussten nur zwei Master hierzu angelegt werden, um das Deckblatt und die restlichen Seiten zu definieren. Da es schon XSL-FO Stylesheets gab, musste hierzu nichts geändert werden und man kam schnell zu seinem Ergebnisdokument. Um weitere Technische Dokumentationen zu testen, wurde das Seitenlayout von einigen weiteren Kundendokumenten betrachtet und über die XSL-FO Entwicklungsumgebung getestet. Diese konnten ohne großen Aufwand und ohne Probleme angelegt und durch die Generierung des FO-Dokuments nach PDF übertragen werden.

7.4 Generierung von Datenblättern

Unter einem Datenblatt versteht man ein Dokument, welches ausschließlich technische Daten eines Produkts beinhaltet. [HETJ98] Für diesen Test wurde ebenso ein Datenblatt aus der internen Datenbank entnommen, um zu überprüfen, ob sich solche Dokumente einfach mit Unterstützung der XSL-FO Entwicklungsumgebung generieren lassen. Für den Test konnte das Standard- XSL-FO Stylesheet verwendet werden, da die Struktur des XML-Dokuments von der Firma FCT verwendet wurde.

Da das Datenblatt nur eine Seite hatte, wurde hierfür nur eine Master-Page verwendet, die recht einfach und schnell zu definieren war. Die Seitensequenz ging ebenfalls recht schnell, da sie nur eine Kopfzeile hatte, die ohne Probleme gesetzt werden konnte. Somit musste nur das passende Template aus dem XSL-FO Stylesheet angezogen werden, um den Inhalt auf die Seite zu bekommen. Das Ergebnisdokument konnte somit einfach und ohne viel Aufwand generiert werden.

7.5 Generierung von Ersatzteilkatalogen

Bei diesem Szenario soll getestet werden, ob ein Ersatzteilkatalog mit der XSL-FO Entwicklungsumgebung generiert werden kann. Bei einem Ersatzteilkatalog handelt es sich um Tabellen mit Bestellinformationen zu Ersatzteilen eines Produkts. Hierzu ist eine Grafik vorhanden, die in viele Einzelteile zerlegt wurde, die jeweils mit Nummern versehen wurden, die auf die entsprechenden Einträge der Tabelle verlinkt sind. Weiterhin ist eine Explosionsdarstellung des Produkts enthalten, die Einzelteile darstellt. Einzelteile sind mit Positionsziffern versehen, die sich auf Einträge in Tabellen beziehen.

Für diesen Test wurde der Ersatzteilkatalog eines Kunden verwendet. Es soll überprüft werden, welche Möglichkeiten für das Seitenlayout bereits umgesetzt werden können bzw. welche noch nicht. Das Seitenlayout des Katalogs ist einheitlich und somit musste nur eine Master-Page angelegt werden. Hierbei hat man schon gemerkt, dass es viele neue Möglichkeiten für die Gestaltung des statischen Inhalts gibt, die mit der Entwicklungsumgebung noch nicht vollständig abgedeckt werden können. Doch die Möglichkeiten sind gegeben und das Ergebnisdokument sah im Vergleich zum Original sehr gut aus.

8 Schlussbetrachtung

In diesem Kapitel sollen nochmals die Vor- und Nachteile der XSL-FO Entwicklungsumgebung zusammengetragen werden, um aufzuzeigen, dass dieser Lösungsansatz einen hohen Vorteilsgrad für den Anwender besitzt. Dabei wurde das Ziel, eine prototypische prozessorunabhängige XSL-FO Entwicklungsumgebung zu realisieren, erfüllt.

Master-Pages

Zusammenfassend kann man sagen, dass die XSL-FO Entwicklungsumgebung einen großen Vorteil für den Anwender bietet. Über die Oberfläche können sehr schnell und vor allem benutzerfreundlich die Seitenvorlagen für die einzelnen Seitenfolgen angelegt werden. Der Anwender ist in seiner Definition so flexibel, dass er sämtliche Seitenformate angeben kann. Vorbelegte Werte ermöglichen es ihm, die passende Auswahl zu treffen. Zur Unterstützung der Größenvorstellung werden ihm die Werte angezeigt, um sich so nochmals ein Bild der Ausmaße zu verschaffen.

Über die Auswahl der Maßeinheiten werden dynamisch die eingegebenen Werte konvertiert. Somit werden ihm Standardwerte vorgegeben bzw. neu berechnet, um so das Handling für ihn so einfach wie möglich zu halten. Da nicht jede Maßeinheit von den FO-Prozessoren unterstützt wird, kann er so bequem diese Einstellung umstellen, ohne die Werte neu eingeben zu müssen. Diese werden durch die Anwendung umgerechnet.

Es können verschiedene Seitenvorlagen zu Sequenzen zusammengefasst werden, um so Unterschiede im Seitenlayout abzubilden, z.B. gerade und ungerade Seiten. In dieser Definition bleiben dem Anwender sämtliche Möglichkeiten offen.

Seitensequenzen

Es gibt eine Vielzahl an Einstellungsmöglichkeiten für eine Seitensequenz. Der Anwender kann selbst festlegen, wieviele Elemente bspw. in der Kopf- oder Fußzeile vorhanden sein sollen und kann diese nach Belieben ändern. Er gelangt schnell zu den passenden Informationen und wird bei der Vergabe der Elemente grafisch unterstützt.

Die Einbindung des XSL-FO Stylesheets für die Darstellung des eigentlichen Seiteninhalts im Druckbereich wurde für den Anwender erfolgreich umgesetzt. Das Stylesheet kann von den Entwicklern der Firma FCT ohne Einschränkungen für den Kunden angepasst werden. Damit sind alle Möglichkeiten gegeben, die mit XSL-FO zur Verfügung gestellt werden. Über die Anwendung kann er nun auf dieses Stylesheet zugreifen und die entsprechenden Methoden aufrufen, um zu bestimmten, welche Inhalte auf die Seiten kommen sollen.

Absatzformate

Beim Anlegen eines neuen Absatzformats werden die auf dem System installierten Schriftarten ermittelt und über die Anwendung für die Definition eines Absatzformats zur Verfügung gestellt. Hierbei wird der Anwender durch die Ansicht des jeweiligen Fonts bei der Auswahl unterstützt, um die Auswirkungen auf den jeweiligen Absatz zu betrachten. Die Auswahl der verschiedenen Schriftarten bietet den Vorteil, dass auch nur die Schriftarten eingesetzt werden können, die auf dem System gefunden wurden. Die Einstellungsmöglichkeiten beschränken sich auf die Wesentlichen. Hierbei könnte man noch einige mehr hinzufügen, um sämtliche Einstellungen, die man für ein Absatzformat treffen kann, für den Anwender zur Verfügung zu stellen.

Im Allgemeinen kann man sagen, dass die XSL-FO Entwicklungsumgebung einen deutlichen Vorteil für den Anwender bringt. Der Anwender hat die Möglichkeit, verschiedene XSL-FO Projekte anzulegen bzw. anzupassen. Dies hat den Vorteil, dass es unterschiedliche Projekte geben kann, die jeweils unterschiedliche Seitenformate und Seitenlayouts beinhalten. Somit ist der Anwender in dieser Hinsicht sehr flexibel.

Durch die verschiedenen Einstellungen, die über die Oberfläche getätigt werden können, kann man recht schnell und einfach das Seitenlayout für die Seitenfolgen festlegen bzw. verändern. Über die Definition der Seitensequenzen können diese Seitenvorlagen einfach referenziert und die Inhalte der Seite festgelegt werden. Die Vorschau bietet dem Anwender die Möglichkeit, die gesetzten Einstellungen über die Generierung eines PDF-Dokuments zu kontrollieren, um so gegebenenfalls Änderungen nachträglich schnell und einfach einstellen zu können.

Die Batchdateien und die XSL-FO Stylesheets, die im Hintergrund erzeugt werden, bieten die Möglichkeit, die Konfigurationen auf andere Server zu portieren. Um das Ergebnisdokument zu generieren, können nun die Batchdateien ausgeführt werden. Somit sind die XSL-FO Stylesheets bereits durch den Anwender konfiguriert worden. Über die XSL-FO Entwicklungsumgebung können verschiedene Projekte verwaltet und angelegt werden, um diese dann nur noch über die Batchprozesse ausführen zu müssen. Dies hat den Vorteil, dass das PDF-Dokument über einen solchen Porzess von einem Server aus generiert werden kann, komplett entkoppelt von der Anwendung.

Was das Ausgeben des eigentlichen Seiteninhalts betrifft, wird es immer minimale Unterschiede in der Darstellung geben. Da jeder FO-Prozessor die XSL-FO Elemente unterschiedlich interpretiert bzw. nicht unterstützt, muss es hierzu verschiedene XSL-FO Stylesheets geben, die auf den Einsatz des jeweiligen FO-Prozessors des Anwenders angepasst werden müssen. Diese XSL-FO Stylesheets beinhalten die eigentlichen Inhaltstemplates, die von einem Entwickler abgeleitet, bzw. angepasst werden.

Natürlich ist diese Anwendung nur ein Prototyp und deckt nicht jede Möglichkeit, die XSL-FO bietet, ab. Um herauszufinden, wie weit die Anwendung in Bezug auf qualitativ hochwertigen Satz gehen kann, wäre es sicher von Vorteil, einen erfahrenen Redakteur damit arbeiten zu lassen. Somit können Optimierungen der Anwendung gefunden und umgesetzt werden.

Ein nicht außer Acht zu lassender Gesichtspunkt wäre ein Element-Mapping mit aufzunehmen. Über ein Element-Mapping könnten verschiedenen Elementen aus der Struktur des Anwenders ein Absatzformat zugewiesen werden. Somit könnten auch die Inhalte, die im Hauptbereich einer Seite durch XSL-FO erzeugt werden, mit einem entsprechenden Absatzformat versehen werden. Dieser Aspekt wurde aus Zeitgründen nicht in den Prototyp mit aufgenommen. Es ist vorstellbar, diese Funktionalität im weiteren Umgang mit der Anwendung zu implementieren.

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Quellenverzeichnis

[@ADOB]	The Seybold Report, Analysing Publbishing Technologies, What is XSL-FO? When should I use it? Auszug vom 19.12.2007 2002, Seybold Publications, ISSN 1533-9211 http://www.adobe.com/de/products/server/documentserver/pdfs/adobeseybold_xsl-fo.pdf
[@APACb]	Apache FOP, Auszug vom 21.09.2007 http://xmlgraphics.apache.org/fop/
[@FCT]	Fischer Computertechnik GmbH, Fischer Computertechnik: XML Redaktionssystem TIM-RS Enterprise-Content-Managementsystem Auszug vom 19.09.2007 http://www.fct.de
[@HP]	HP Deskjet-Drucker - Verwenden der Druckersprache PCL, Auszug vom 24.09.2007 http://h10025.www1.hp.com/ewfrf/wc/genericDocument?lc=de&cc=de&docname=bgd09160
[@MIDH]	Antenna House XSL Formatter - Produktbeschreibung Auszug vom 24.09.2007 http://www.mid-heidelberg.de/produkte/formatierung/antenna_01.htm
[@REND]	XEP User Guide, Auszug vom 24.09.2007 http://www.renderx.com/tools/xep.html
[@SVG]	W3C Recommendation, 1.1 About SVG Auszug vom 24.09.2007 http://www.w3.org/TR/2003/REC-SVG11-20030114/intro.html
[@W3SC]	W3Schools, XSL-FO writing-mode Property Auszug vom 28.09.2007 http://www.w3schools.com/xslfo/prop_writing-mode.asp
[GAMMA96]	Entwurfsmuster, Elemente wiederverwendbarer objektorientierter Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 1996 by Addison Wesley Verlag, ISBN-3-8273-1862-9
[KRUE06]	XSL-FO verstehen und anwenden, XML-Verarbeitung für PDF und Druck, Manfred Krüger, 2006 dpunkt.verlag GmbH ISBN 3-89864-394-8

[HETJ98]	Wörterbuch zur technischen Kommunikation und Dokumentation, Jörg Henning, Marita Tjarks-Sobhani, 1998 Verlag Schmidt-Römhild, Lübeck ISBN 3-7950-0740-2
[PIKR04]	XSL-FO in der Praxis, XML-Verarbeitung für PDF und Druck, Manuel Montero Pineda, Manfred Krüger, dpunkt.verlag GmbH, 1. Auflage 2004 ISBN 3-89864-249-6
[WAES06]	E-Business, Technologien und Anwendungen, Prof. Dr.-Ing. Jürgen Wäsch, Fachhochschule Konstanz \\merkur\lehre\Waech\EBUS\Folienskriptum_Vorlesung_WS0607\02 Technologische Grundlagen EBWI.pdf

A Anhang

An dieser Stelle wird die Benutzeroberfläche der XSL-FO Entwicklungsumgebung vorgestellt. Es soll ein Eindruck vermittelt werden, wie man mit der Anwendung umgehen kann, um einfach und schnell zu einem PDF-Dokument zu gelangen. Anschließend werden noch einige Tools und Technologien vorgestellt, die bei dieser Diplomarbeit verwendet wurden.

A.1 Anwendung der XSL-FO Entwicklungsumgebung

In diesem Kapitel soll nun darauf eingegangen werden, welche Angaben der Anwender treffen kann, um einfach und bequem und vor allem schnell zu seinem Ziel zu gelangen.

Für die Beschreibung der Benutzeroberfläche soll ein neues Projekt angelegt werden. Wenn der Anwender die Anwendung startet, wird ihm über ein Menü die Wahl angeboten, vorhandene XSL-FO Projekte zu bearbeiten oder neue FO-Projekte zu erstellen.

In diesem Teil der Diplomarbeit wird die Benutzerfreundlichkeit anhand des Erstellens eines Deckblatts dargestellt.

A.1.1 Anlegen eines neuen XSL-FO Projekts

Der Anwender gelangt über den Menüeintrag ***Datei > Projekt anlegen*** zu folgender Darstellung.

Fig. 11-1 : Neues XSL-FO Projekt anlegen

Der Anwender muss nun hier den Namen des Projekts, den Speicherort, die Konfigurationsdatei (Einstellungen für die FO- und XSLT- Prozessoren) sowie die jeweiligen Konfigurationsdateien der Master-Pages, Seitensequenzen und Absatzformate festlegen. Dabei ist wichtig, dass alle Angaben eingegeben werden müssen. Es ist von großem Vorteil, wenn der Anwender bereits ein Verzeichnis angelegt hat, welches für die FO-Projekte benutzt werden kann. In der Konfigurationsdatei müssen lediglich nur die Pfade angepasst werden. Wenn alle Parameter angegeben worden sind kann über ***Fertigstellen*** das Projekt unter dem angegebenen Speicherort abgelegt und geladen werden.

A.1.2 Oberfläche zur Definition der Projekteinstellungen

Wenn das Projekt geladen wird, so gelangt der Anwender zur Projektoberfläche. Über diese Oberfläche gelangt der Anwender schnell zu den Einstellungsmöglichkeiten der Master-Pages, Seitensequenzen, Absatzformate und des aktuellen Projekts. Die nachkommende Abbildung zeigt die Ansicht genauer.

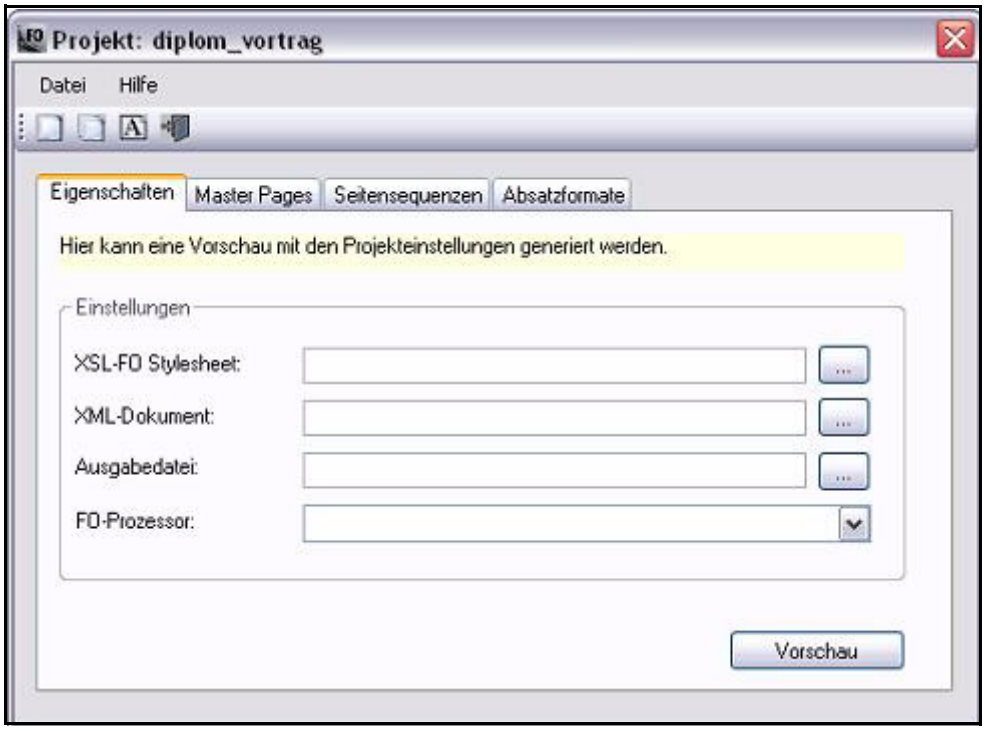


Fig. 11-2 : Projektansicht mit Überblick über Projekt-Einstellungen

Für den schnellen Zugriff auf die Bereiche Eigenschaften, Master-Pages, Seitensequenzen und Absatzformate wurden Registerkarten definiert. Eine andere Möglichkeit wäre gewesen, diese Funktionalität in das Menü mit einzugliedern, jedoch wäre dies nicht so schnell für den Anwender zu erreichen gewesen. Jetzt wird nur ein Klick auf das jeweilige Register benötigt, was die Usability enorm steigert. Auch über Buttons in der Symbolleiste können die Hauptfunktionen "Neue Master-Page", "Seitensequenz" und "Neues Absatzformat anlegen" schnell erreicht werden.

A.1.3 Oberfläche zur Bearbeitung von Master-Pages

Wie bereits im Ablaufdiagramm Fig. 5-8, Seite 51 geschildert wurde, ist der erste Schritt für den Anwender nun, eine Seitenvorlage für das Deckblatt zu definieren. Hierzu muss der Anwender nun auf das Register ***Master-Pages*** wechseln (siehe Abbildung Fig. 11-3). Bereits vorhandene Master-Pages werden in einer Liste angezeigt. Diese können markiert werden und über den Button ***Bearbeiten*** geändert oder über ***Löschen*** aus dem Konfigurationsfile entfernt werden. In dieser Ansicht kann der Anwender ebenso neue Master-Pages definieren. Dies kann er über den Button ***Erstellen***.

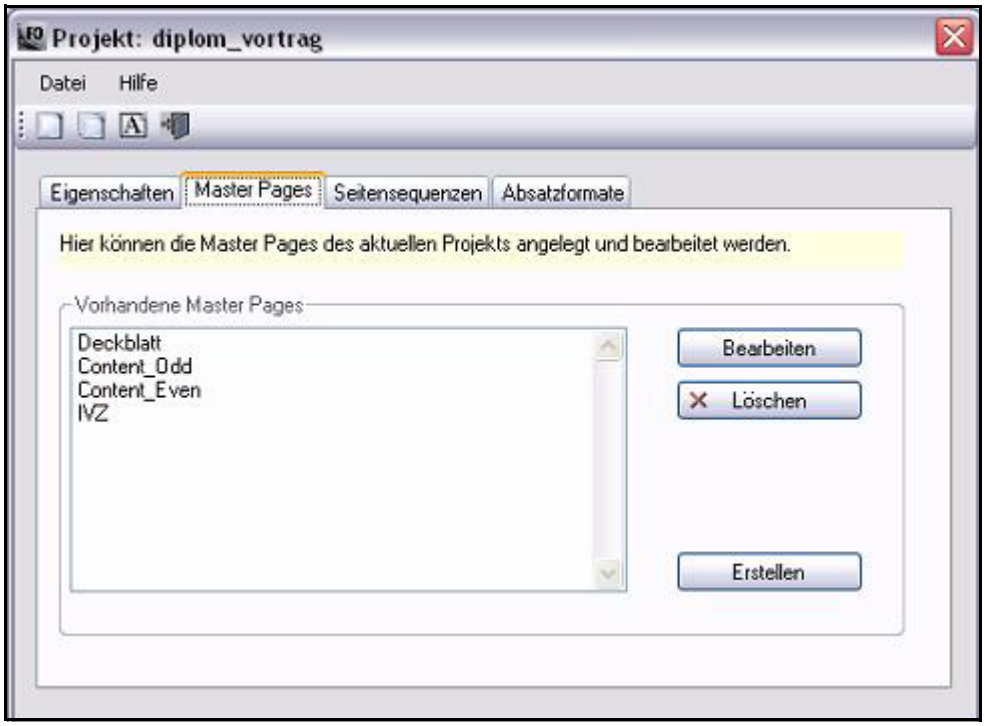


Fig. 11-3 : Register zur Bearbeitung und Erstellung von Master-Pages

Über den Button ***Erstellen*** wird nun eine neue Master-Page angelegt. Die Abbildung Fig. 11-4 stellt diese Oberfläche dar. Hier kann der Anwender nun seine Master-Page konfigurieren. Pflichtfelder sind durch einen Stern eindeutig gekennzeichnet. Der Name einer Master-Page muss dabei unbedingt angegeben werden, da über diesen später beim Anlegen der Seitensequenz dieser Master referenziert wird.

Bei der Definition der Seitenränder werden die möglichen Maßeinheiten in einer Auswahlliste angeboten. Diese Auswahlmöglichkeit an Maßeinheiten bietet dem Anwender den Vorteil, dass er hierzu nur noch den jeweiligen Wert in das Eingabefeld eintragen muss, ohne jedesmal dahinter die Einheit schreiben zu müssen. Sollte sich der Anwender für eine andere Maßeinheit entscheiden, so wird der Wert, der zuvor gesetzt wurde, neu berechnet und im Eingabefeld aktualisiert.

Bei den Einstellungen einer Master-Page kann ebenso das Seitenformat mit angegeben werden. Hierzu gibt es ebenso eine Auswahlliste mit den entsprechenden gängigen Formaten, bspw. *A4, A2, A0*, etc. Sollte dem Anwender eines dieser Formate nicht zusprechen, so hat er die Möglichkeit, über den Eintrag *Benutzerdefiniert* einen neuen Wert zu vergeben. Bei dieser Auswahl wird ihm eine kleine Oberfläche zur Verfügung gestellt, die die Eingabefelder der Höhe und Breite der Seite beinhaltet.

Unter der Auswahlliste der Seitenformate wird ihm jedesmal das entsprechende Format mit angezeigt, bspw. für A4 die Werte 210mm * 297mm. Somit hat er nochmals die Gewissheit, dass er auch das richtige Format ausgewählt hat.

Da eine Seite nicht immer im Hochformat dargestellt werden kann, gibt es hier die Möglichkeit, ebenso die Seite als Querformat anzeigen zu lassen. Mit einem kleinen Bild wird der Eindruck der Ausprägung noch deutlicher.

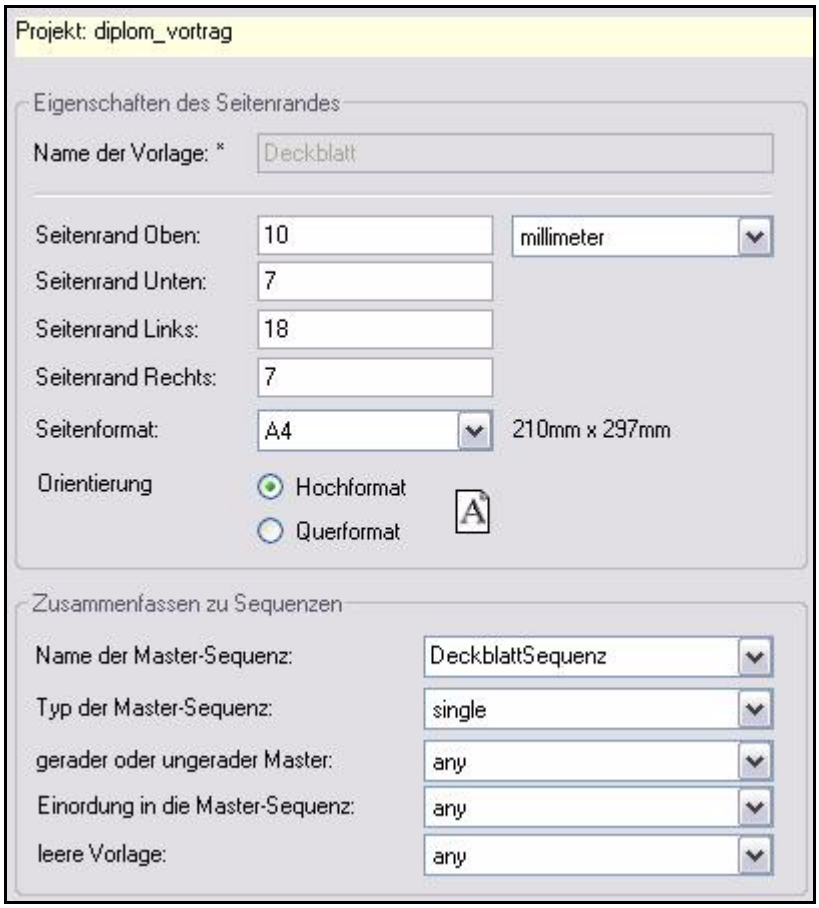


Fig. 11-4 : Einstellungen einer Master-Page

Nun ist es möglich, eine Master-Page nicht nur für eine Seite zu bestimmen, sondern auch für fortlaufende Seiten zu verwenden. Hierzu kann eine Master-Page einer Sequenz zugeordnet werden, die verschiedene Master-Pages beinhaltet. Dies ist vor allem ein Vorteil, wenn es darum geht, ungerade oder gerade Seiten zu definieren. Zunächst muss hier ein Name für die Sequenz vergeben werden. Einmal angelegte Sequenzen werden in einer Auswahlliste verfügbar gemacht und müssen nicht neu definiert werden, wenn eine neue Master-Page erstellt wird.

Weiterhin können der Typ der Sequenz, ob dieser Master sich auf gerade oder ungerade Seiten bezieht, wo dieser Master in der Sequenz eingeordnet werden und ob dieser Master eine leere Seiten sein soll angegeben werden.

Die Benutzeroberfläche für die Konfiguration der Seitenbereiche wird in der nachfolgenden Abbildung gezeigt. Für diese Konfiguration bieten sich wiederum Registerkarten an, über die schnell zu den einzelnen Bereichen gesprungen werden kann.

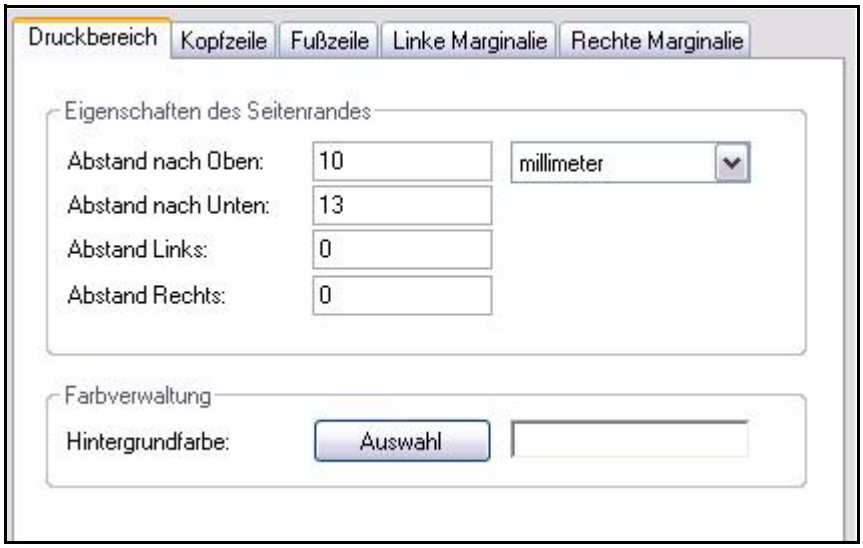


Fig. 11-5 : Konfigurationen der einzelnen Bereiche einer Master-Page

Hier kann der Anwender die Einstellungen für die Seitenränder angeben. Auch für diese Einstellungen werden Standardwerte vorbelegt. Wenn alle Werte gesetzt wurden, kann die Master-Page über den Button ***Speichern*** in das Konfigurationsfile der Master-Pages gespeichert werden. Sollte die Master-Page doch nicht angelegt werden, kann der Prozess über den Button ***Abbrechen*** abgebrochen werden.

Nach dem Speichern gelangt der Anwender wieder in die Ansicht Master-Pages (siehe Abbildung Fig. 11-3) und sieht den definierten Master in der Auswahlliste. Somit hat der Anwender eine Seitenvorlage geschaffen, die er für die Seitensequenzen verwenden kann.

A.1.4 Oberfläche zur Bearbeitung von Seitensequenzen

Wenn der Anwender nun seine Vorschau erstellen möchte, so wird er in diesem Fall noch nichts sehen, da die Seitensequenz noch fehlt. Um eine Seitensequenz anzulegen, wird auf das Register ***Seitensequenzen*** gewechselt. Diese Darstellung unterscheidet sich kaum vom der einer Master-Page. Zusätzlich kann die reihenfolge der aufeinander folgenden Seitensequenzen verändert werden. Die nachfolgende Abbildung zeigt wie dies gemeint ist.

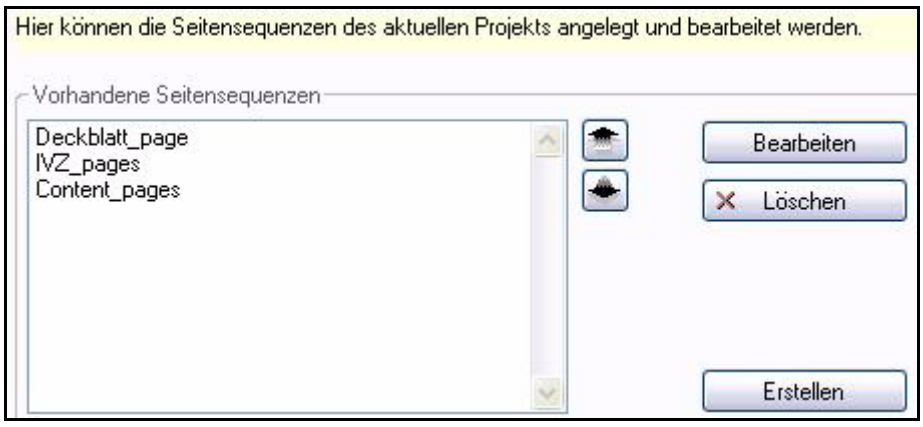


Fig. 11-6 : Register zur Bearbeitung und Erstellung einer Seitensequenz

Hier gibt es zwei Richtungspfeile, über die der Anwender bequem die ausgewählte Seitensequenz nach oben, bzw. nach unten hin verschieben kann. Die Reihenfolge der Sequenzen in der Anwendung entspricht der späteren Reihen-

folge der Inhalte im PDF. Wird eine neue Seitensequenz erstellt, wird folgende Darstellung geöffnet.

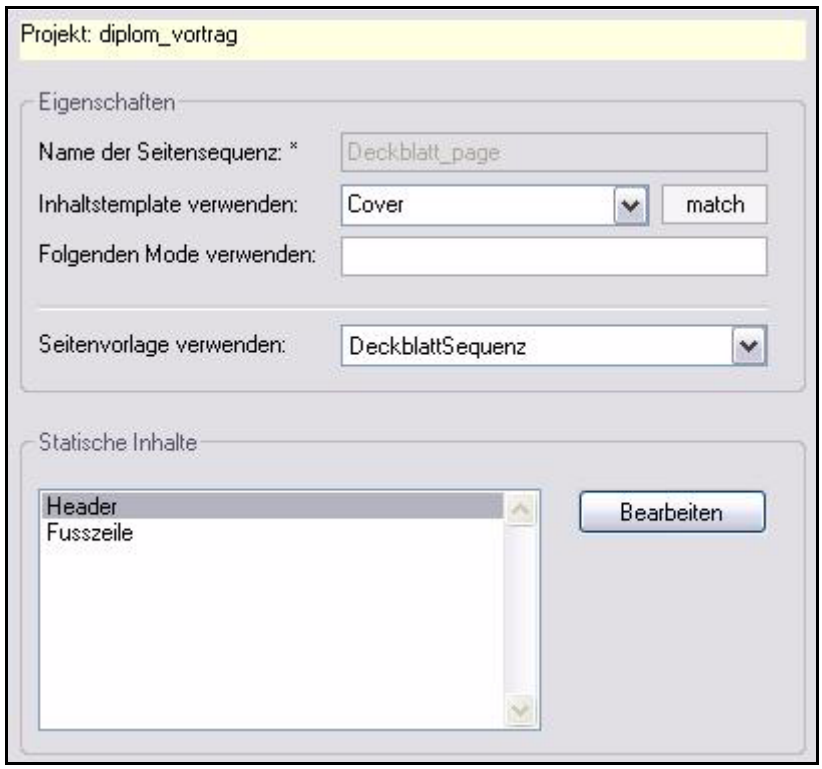


Fig. 11-7 : Seitensequenzen anlegen und Master-Page referenzieren

Hier kann der Anwender nun eine neue Seitensequenz anlegen, indem er einen geeigneten Namen vergibt. Im Feld "Seitenvorlage verwenden" werden über eine Auswahlliste die zuvor angelegten Master-Page Sequenzen dargestellt. Bei der Auswahl einer Sequenz werden im unteren Teil des Fensters die jeweiligen Bereiche angezeigt, die mit statischen Inhalten befüllt werden können. Über den Button ***Bearbeiten*** gelangt man zu einer weiteren Oberfläche, über die der Anwender auswählen kann, was für Elemente in den jeweiligen Bereich einzusetzen sind.

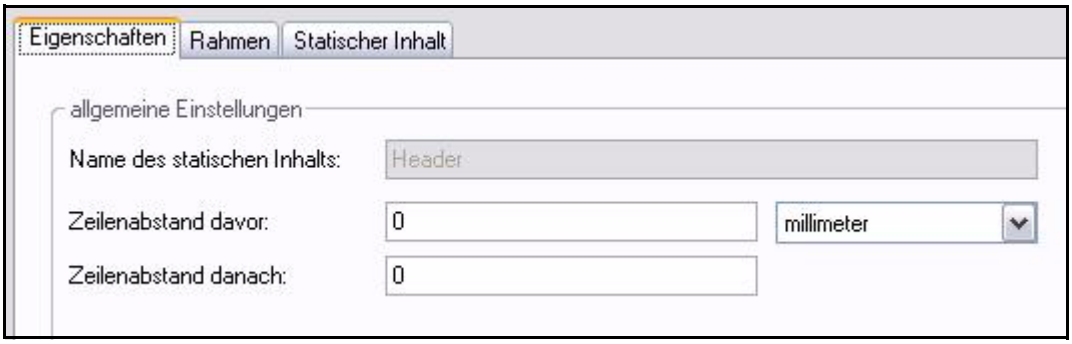


Fig. 11-8 : Einstellungsmöglichkeiten für statische Inhalte

Nachdem der Anwender nun einen Bereich ausgewählt hat, kann er spezielle Einstellungen für diesen tätigen. Zu diesen Einstellungsmöglichkeiten zählt der Zeilenabstand des jeweiligen Inhalts des ausgewählten Bereichs nach oben bzw. nach unten. Auch hier kann die Maßeinheit über eine Auswahlliste gewählt werden.

Im Rahmen kann eine Linie für den Bereich definiert werden, bspw. eine Linie für die Kopfzeile.

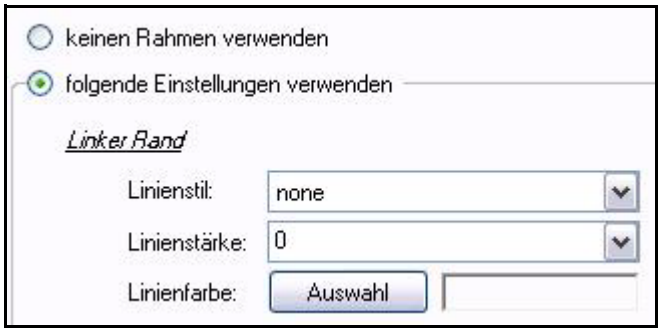


Fig. 11-9 : Einstellungen für den Rand eines Bereichs

Linien können für den oberen, unteren, linken und rechten Rand definiert werden. Zu den Einstellungsmöglichkeiten eines Rahmens gehört der Linienstil, die Linienstärke und die Linienfarbe. Bei der Auswahl einer Farbe, wird dem Anwender eine Farbpalette angeboten, wobei er auch selbst Farben frei definieren kann.

Um nun bspw. eine Überschrift oder eine Grafik in die Kopfzeile zu platzieren, wird auf das Register ***Statische Inhalte*** gesprungen.

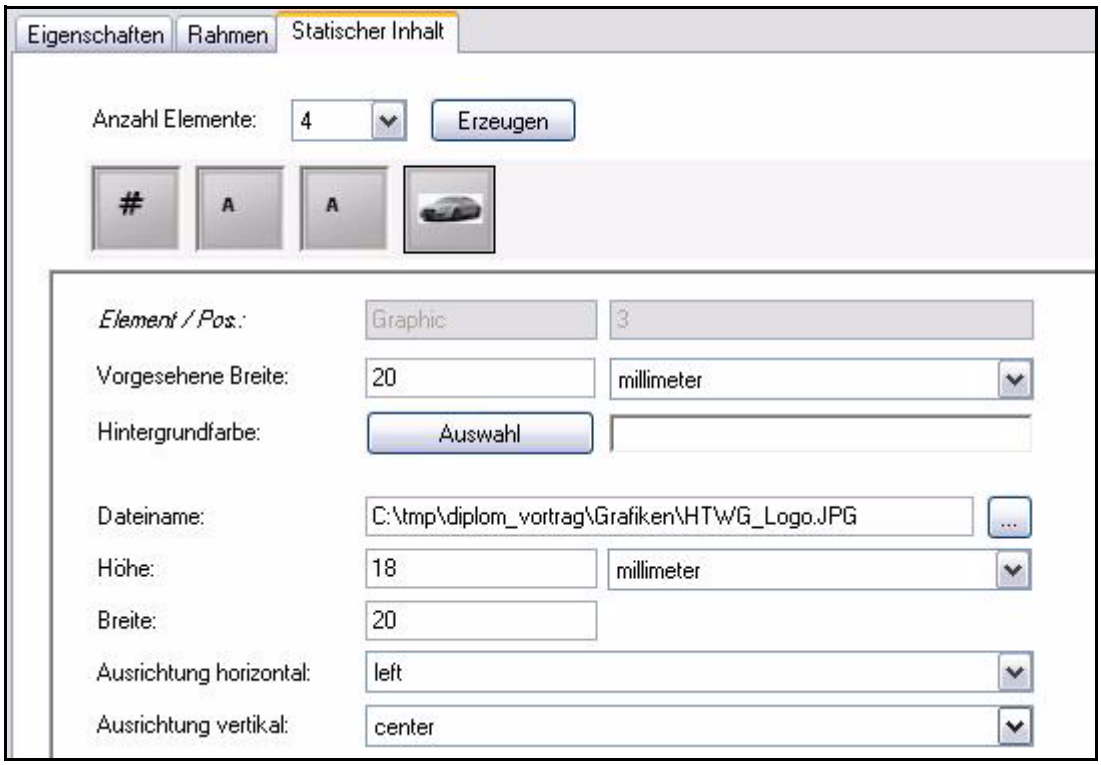


Fig. 11-10 : Elemente auswählen und bearbeiten

Hier hat der Anwender nun die Möglichkeit anzugeben, wieviele Elemente er für den statischen Inhalt in seinem Bereich platzieren möchte. Es wird ihm eine Auswahl angeboten, die er dann über Drag and Drop in die jeweiligen Bereiche ziehen kann. Hierbei kann dieser Abschnitt (bspw. Kopfzeile) in bis zu maximal acht Bereiche unterteilt werden. Das bedeutet, dass bis zu acht Elemente in diesem Bereich platziert werden können. Hier gibt es die Auswahl zwischen Grafik- , Zeichenfolgen- , Überschrift- und Seitenzahlelementen. Durch Klicken auf das jeweilige Element werden die dazu gehörigen Informationen direkt unter der Elementliste angezeigt. Im oberen Beispiel wurde die Grafik am rechten Seitenrand selektiert. Hier können nun die Informationen verändert und über ***Überneh-***

men für das Element gespeichert werden. Somit konfiguriert der Anwender nun seine Elemente und speichert diese für den statischen Bereich ab.

Nachdem der Anwender diesen Bereich gespeichert hat, gelangt er wieder zurück zu den Einstellungen einer Seitensequenz. Diese Einstellungen betreffen nur die statischen Inhalte für die Bereiche Kopfzeile, Fußzeile, linker und rechter Marginalie.

Um nun den Inhalt in die Seitensequenz zu bekommen, muss ein Inhaltstemplate aus dem angegebenen XSL-FO Stylesheet ausgewählt werden können.

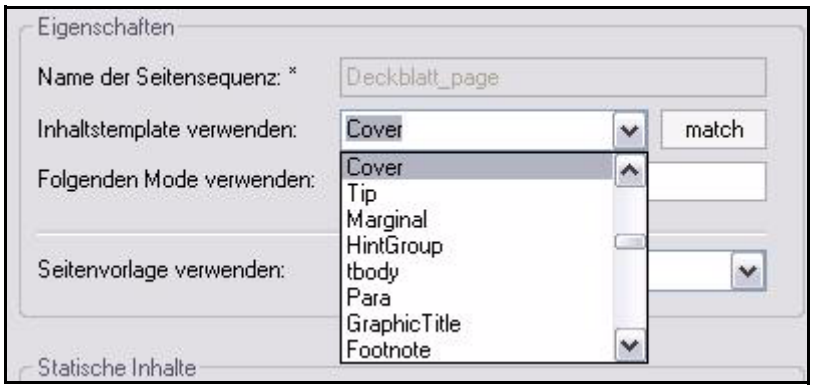


Fig. 11-11 : Template für den Inhalt einer Seite auswählen

Hierbei wird dem Anwender eine Liste mit den verfügbaren Inhaltstemplates angeboten. Somit sollte er wissen, welches Template welche Auswirkung auf den Inhalt der Seite haben wird. Nun kann das PDF-Dokument und somit das FO-Dokument erstellt werden. Dazu speichert der Anwender die Seitensequenz ab und gelangt wieder in das Register Seitensequenzen der Projektoberfläche.

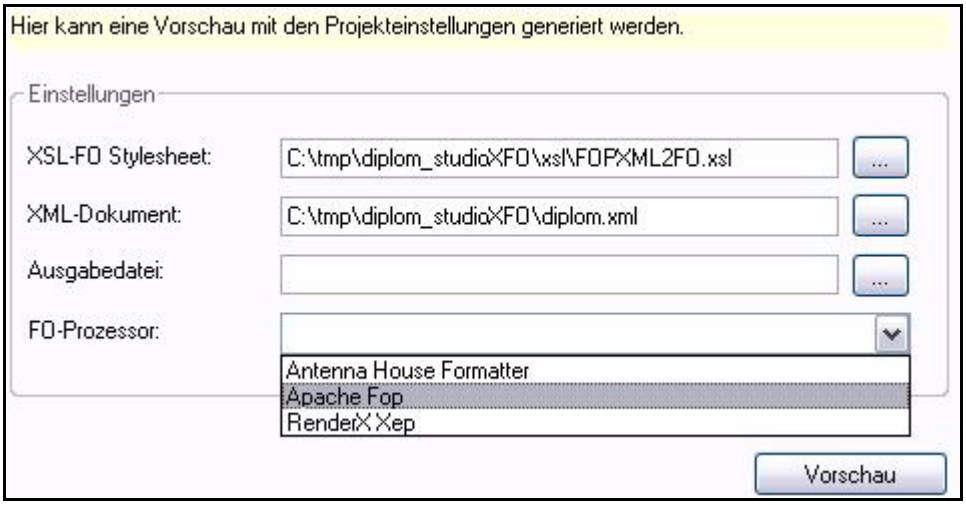


Fig. 11-12 : Vorbereitung auf die Vorschau eines FO-Dokuments

Damit nun die Vorschau generiert werden kann, muss ein entsprechender FO-Prozessor ausgewählt werden. Über den Button ***Durchsuchen*** können aus dem System das XSL-FO Stylesheet, das XML-Dokument (für die Darstellung der Inhalte) und die Ausgabedatei (das FO-Dokument) ermittelt werden. Um den Prozess zu starten wird der Button ***Vorschau*** betätigt. Hier spielt wieder die Benennung der Buttons eine große Rolle, damit dem Anwender sofort klar wird, welche Funktionalität dahinter steckt. Ein entsprechender Fortschrittsbalken unterstützt den Anwender dabei, um festzustellen, wie weit die Überführung des Ergebnisdokuments ist. Wenn alles korrekt war, öffnet sich das PDF-Dokument für den Anwender. Somit kann er überprüfen, ob seine Einstellungen auch seinen

Wünschen entsprechen, um im Nachhinein Änderungen an den Master-Pages oder Seitensequenzen vorzunehmen.

A.1.5 Oberfläche zur Bearbeitung von Absatzformaten

Die Anwendung bietet dem Anwender die optionale Möglichkeit, Absatzformate zu definieren. Hierzu gibt es das Register ***Absatzformate***, über welches Absatzformate erstellt, entfernt und bearbeitet werden können. Wie auch bei den anderen Registern gibt es eine Auswahl an vorhandenen Absatzformaten. Somit hat der Anwender immer einen Überblick über die in der Konfiguration gesicherten Objekte. In diesem Beispiel wird ein neues Absatzformat angelegt, das für die statischen Inhalte, als auch für das XSL-FO Stylesheet verwendet werden kann. Die nachfolgende Abbildung zeigt die Oberfläche.

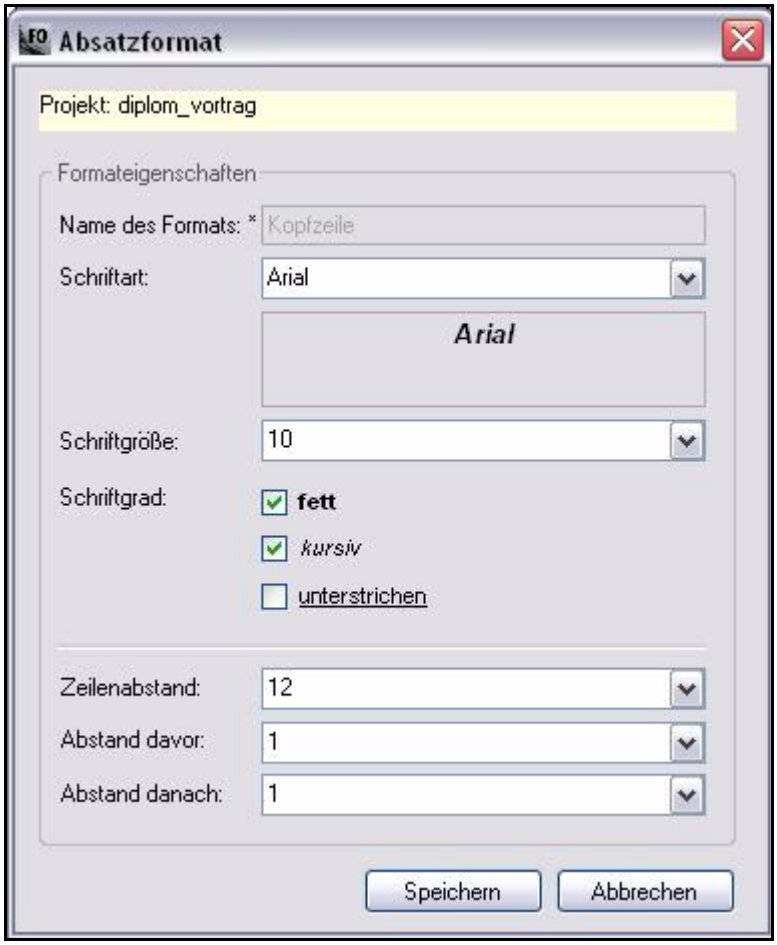


Fig. 11-13 : Parameterwahl für Absatzformate

Wenn ein neues Absatzformat erstellt wird, werden gewisse Parameter schon vorbelegt. Beispielsweise kann eine gewisse Schriftart (im oberen Beispiel Arial) schon gesetzt werden, ebenso wie eine Schriftgröße und Zeilenabstände. Um ein Format im System zu hinterlegen, muss ein geeigneter Name vergeben werden. Bei der Auswahl einer Schriftart werden vorhandene Schriftarten, die auf dem System des Anwenders installiert sind, erkannt und in einer Liste dargeboten. Da nicht alle FO-Prozessoren dieselben Namen dieser Schriftarten unterstützen, kann der Name hierfür einfach in das Feld eingetragen werden. Damit der Anwender sich ein Bild über die Schriftart machen kann, wird unter dieser Auswahlliste ein kleines Fenster angezeigt, das den ausgewählten Font abbildet. Ebenso gibt es Auswahllisten für die Schriftgröße, Zeilenabstände und Leerräume zwischen den einzelnen Zeichenfolgen. Außerdem werden dem Anwender

Checkboxen angeboten, die es erlauben einen Zeichenfolge gleichzeitig fett, kursiv und unterstrichen zu gestalten. Um das Absatzformat in der Konfigurationsdatei zu speichern, wird dieser Prozess über den Button ***Speichern*** eingeleitet. Über ***Abbrechen*** kann diese Aktion jederzeit abgebrochen und zur Projektoberfläche zurückgekehrt werden.

A.2 Eingesetzte Technologien und Anwendungen

In diesem Kapitel der Diplomarbeit soll kurz aufgelistet werden, welche Technologien und Anwendungen für das Entwickeln dieser prozessorunabhängigen XSL-FO Entwicklungsumgebung eingesetzt wurden.

A.2.1 Microsoft Visual Studio 2005 (C#)

Zur Entwicklung der Applikation wurde die Programmiersprache C# eingesetzt. Dabei wurde mit der Entwicklungsumgebung Microsoft Visual Studio 2005 gearbeitet.

A.2.2 Microsoft Office Visio Professional 2003

Zur Erstellung der verschiedenen Use-Cases und Klassendiagramme wurde Microsoft Office Visio eingesetzt.

A.2.3 Stylus Studio 2007 XML Professional Suite

Um die XML-Dokumente mit den Konfigurationsparametern der Master-Pages und Seitensequenzen zu testen, wurden diese über die Anwendung Stylus Studio 2007 erfasst und bearbeitet. Ebenso zur Entwicklung der XSL-FO Stylesheets wurde die Professional Suite eingesetzt. Gerade in der Konzeptionsphase kam diese Anwendung zum Einsatz, um zu testen, ob der Weg zum Ziel sich dadurch bewerkstelligen lässt.

A.2.4 Antenna House XSL Formatter

Um aus dem FO-Dokument ein PDF-Dokument generieren zu können, wurde der FO-Prozessor Antenna House XSL Formatter V4.2 verwendet. Hierbei handelte es sich um eine Testversion.

A.2.5 RenderX XEP

Neben dem Antenna House XSL Formatter wurde der FO-Prozessor RenderX XEP V4.10 verwendet, um aus dem generierten FO-Dokument ein PDF-Dokument zu erzeugen. Hierbei handelte es sich um eine Testversion.

A.2.6 Apache FOP

Die Kunden von FCT setzten bei der bisherigen XSL-FO Umsetzung den FO-Prozessor FOP V0.20.5 zur Erzeugung ihrer PDF-Dokumente ein. Dieser konnte ohne Einschränkungen für den Test der XSL-FO Entwicklungsumgebung eingesetzt werden.

A.2.7 Adobe FrameMaker

Die Diplomarbeit wurde mit dem Strukturedtor Adobe FrameMaker erfasst.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, Daniel Merkle, geboren am 26.02.1984 in Radolfzell,

(1) dass ich meine Diplomarbeit mit dem Titel:

"Analyse, Konzeption und prototypische Umsetzung einer prozessorunabhängigen XSL-FO Entwicklungsumgebung"

bei der Firma Fischer Computertechnik GmbH unter Anleitung von Prof. Dr. Heiko von Drachenfels selbständig und ohne fremde Hilfe angefertigt habe und keine anderen als die angeführten Hilfen benutzt habe;

(2) dass ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 16.01.2008

Daniel Merkle